Phần 1

ÔN TẬP
PHẦN THI BUỔI SÁNG

Các câu hỏi trong phần thi buổi sáng nằm trong bảy lĩnh vực sau: Khoa học máy tính cơ sở, hệ thống máy tính, phát triển hệ thống, công nghệ mạng, công nghệ cơ sở dữ liệu, bảo mật và chuẩn hóa, tin học hóa và quản lý.
Phần đầu của mỗi chương sẽ giải thích chi tiết về mỗi lĩnh vực trên, tiếp theo là các câu hỏi thực tế đã được sử dụng trong các bài thi trước đây, các câu trả lời và giải thích nằm ở cuối mỗi chương.

1 Khoa học máy tính cơ sở

Mục tiêu của chương này

Để trở thành một kĩ sư công nghệ thông tin, cần phải hiểu cấu trúc của thông tin được xử lí bởi máy tính và ý nghĩa của quá trình xử lý thông tin. Tất cả thông tin được lưu trữ trong máy tính ở dạng số nhị phân; do đó trong phần 1, ta sẽ nghiên cứu về dạng mà số thập phân và kí tự sử dụng trong cuộc sống hàng ngày được lưu trữ trong máy tính. Trong phần 2, ta sẽ nghiên cứu về các phép toán logic qua các ví dụ cụ thể của quá trình xử lý thông tin. Trong phần 3, ta sẽ nghiên cứu về các cấu trúc dữ liệu mà sự biến đổi trên đó là cần thiết để quá trình xử lý dữ liệu dễ dàng hơn. Cuối cùng, trong phần 4, ta sẽ nghiên cứu về các phương pháp xử lý dữ liệu cụ thể.

1.1 Nguyên lý cơ bản về thông tin
1.2 Thông tin và logic
1.3 Cấu trúc dữ liệu
1.4 Giải thuật

[Thuật ngữ và khái niệm cần nắm vững]
Cơ số, nhị phân, hệ 16, dấu phẩy cố định, dấu phẩy động, tổng logic, tích logic, tổng loại trừ logic, bộ cộng, danh sách, ngăn xếp, hàng đợi, tìm kiếm

tuyến tính, tìm kiếm nhị phân, sắp xếp nổi bọt

guyên lý cơ bản về thông tin

Mở đầu

Tất cả thông tin (kí tự và số) được biểu diễn trong máy tính bởi sự kết hợp của các kí tự 0 và 1. Một biểu diễn chỉ sử dụng các kí tự 1 và 0 được gọi là 1 số nhị phân. Trong phần này, ta sẽ học về dạng biểu diễn thông tin

1.1.1 Chuyển đổi cơ số

Điểm chính	➢ Trong máy tính, tất cả dữ liệu được biểu diễn bởi các số nhị phân ➢ Các số hệ 16 được biểu diễn bằng cách tách các số nhị phân thành các nhóm 4-bit.

Thuật ngữ "Chuyển đổi cơ số"[1] nghĩa là, ví dụ, chuyển một số thập phân thành một số nhị phân. Ở đây "10" trong số thập phân và "2" trong số nhị phân được gọi là các cơ số. Trong máy tính tất cả dữ liệu được biểu diễn dưới dạng **số nhị phân** tương ứng với 2 trạng thái điện ON và OFF. Mỗi chữ số của một số nhị phân chỉ có thể là "0" hoặc "1", nên tất cả các số được biểu diễn bởi 2 kí tự 0 và 1.

Tuy nhiên, các số nhị phân biểu diễn bởi sự kết hợp của các kí tự 0 và 1 dài và khó hiểu, nên khái niệm **hệ cơ số 16** được đưa ra. Trong hệ cơ số 16, 4 bit[2] (tương ứng với các số từ 0 đến 15 trong hệ thập phân) được biểu diễn bởi 1 chữ số (0..9, A..F)

Bảng sau chỉ ra sự tương ứng giữa hệ thập phân, hệ nhị phân, và hệ cơ số 16.

Số thập phân	Số nhị phân	Số hệ 16	Số thập phân	Số nhị phân	Số hệ 16
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B

[1] **Cơ số**: là số tạo ra trọng số của mỗi chữ số trong hệ số như nhị phân, hệ 8, hệ thập phân, hệ 16. Cơ số tương ứng của các hệ số là 2, 8, 10, 16.
Hệ nhị phân: sử dụng 0 và 1
Hệ cơ số 8: sử dụng từ 0 đến 7
Hệ thập phân: sử dụng từ 0 đến 9
Hệ cơ số 16: sử dụng từ 0 đến F

[2] **Bit**: đơn vị thông tin nhỏ nhất trong 1 máy tính, biểu diễn bởi "0" hoặc "1". Dữ liệu trong máy tính biểu diễn trong dạng nhị phân, 1 bit biểu diễn 1 chữ số trong hệ nhị phân. Để thuận tiện, số hệ 16 và hệ 8 được biểu diễn bởi phân chia số nhị phân như sau:
Hệ 4: nhóm 2 bit (biểu diễn bởi các chữ số từ 0 đến 3)
Hệ 8: nhóm 3 bit (biểu diễn bởi các chữ số từ 0 đến 7)
Hệ 16: nhóm 4 bit (biểu diễn bởi các chữ số từ 0 đến F)

4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F
			16	10000	10

◆ Chuyển số nhị phân và số hệ 16 thành số thập phân

Tổng quát, khi một giá trị đưa ra trong hệ đếm với cơ số r (hệ cơ số r), ta nhân giá trị mỗi chữ số với **trọng số**[3] tương ứng và cộng các tích lại để lấy giá trị trong hệ thập phân. Với các chữ số từ bên trái của dấu phẩy, trọng số là r^0, r^1, r^2, ... từ chữ số thấp nhất. Phép chuyển đổi được trình bày như sau. (trong các ví dụ này, (a) biểu diễn trong hệ 16 và (b) là trong hệ nhị phân)

$(12A)_{16} = 1 \times 16^2 + 2 \times 16^1 + A \times 16^0$
$= 256 + 32 + 10$
$= (298)_{10}$ ..... (a)

$(1100100)_2 = 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
$= 64 + 32 + 4$
$= (100)_{10}$ ..... (b)

Với các chữ số ở bên phải của dấu phẩy, trọng số lần lượt là r^{-1}, r^{-2}, r^{-3}, ... Nên, phép chuyển đổi được trình bày như sau. Trong các ví dụ này, (c) biểu diễn trong hệ 16 và (d) là trong hệ nhị phân.

$(0.4B)_{16} = 4 \times 16^{-1} + B \times 16^{-2}$
$= 4 / 16 + 11 / 16^2$
$= 0.25 + 0.04296875$
$= (0.29296875)_{10}$ ..... (c)

$(0.01011)_2 = 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5}$
$= 0.25 + 0.0625 + 0.03125$
$= (0.34375)_{10}$ ..... (d)

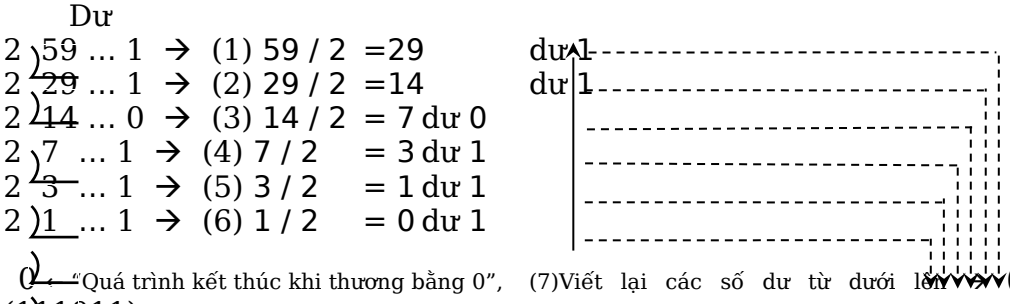
◆ Chuyển số thập phân nguyên thành số nhị phân

Một cách toán học, sử dụng đặc điểm chữ số thứ n từ bên phải (thấp nhất) trong hệ nhị phân biểu diễn sự có mặt của giá trị 2^{n-1}, ta có thể tách số thập phân thành tổng các lũy thừa của 2 (giá trị 2^n cho n).

$(59)_{10} = 32 + 16 + 8 + 2 + 1 = 2^5 + 2^4 + 2^3 + 2^1 + 2^0$
$= 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$
$(1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1)_2$

[3] **Trọng số**: trọng số, giá trị xác định tỉ lệ theo vị trí trong các biểu diễn số, như nhị phân, 8, 10 và 16

Tuy nhiên, ta cũng có thể chia số đã cho liên tiếp cho 2 cho đến khi thương bằng 0. Đây là một phương pháp chuyển đổi máy móc giúp giảm bớt sai số tính toán. [4]


Dư
2) 59 ... 1 → (1) 59 / 2 =29
2) 29 ... 1 → (2) 29 / 2 =14
2) 14 ... 0 → (3) 14 / 2 = 7 dư 0
2) 7 ... 1 → (4) 7 / 2 = 3 dư 1
2) 3 ... 1 → (5) 3 / 2 = 1 dư 1
2) 1 ... 1 → (6) 1 / 2 = 0 dư 1
0 ← "Quá trình kết thúc khi thương bằng 0", (7)Viết lại các số dư từ dưới lên
dư 1
dư 1

$(59)_{10} = (111011)_2$

Thêm nữa, để chuyển 1 số thập phân thành số hệ 16, ta có thể sử dụng 16 thay cho 2. Tổng quát, để chuyển một số thập phân thành số hệ cơ số *n*, dùng *n* thay thế cho 2.

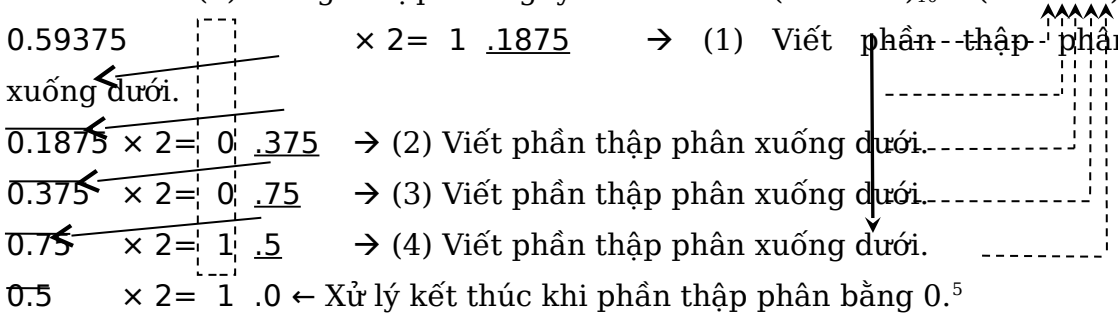
◆ Chuyển số thập phân thành số nhị phân

Một cách toán học, sử dụng đặc điểm chữ số thứ n sau dấu phẩy trong hệ nhị phân biểu diễn sự có mặt của giá trị 2^{-n}, ta có thể tách số thập phân thành tổng các lũy thừa của 2 (giá trị 2^n cho *n*)

$$(0.59375)_{10} = 0.5 + 0.0625 + 0.03125$$
$$= 2^{-1} + 2^{-4} + 2^{-5}$$
$$= 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5}$$
$$(0.1 \quad 0 \quad 0 \quad 1 \quad 1)_2$$

Tuy nhiên, ta có thể nhân phần thập phân (phần bên phải của dấu phẩy) liên tiếp 2 đến khi phần thập phân bằng 0. Đây là phương pháp chuyển đổi máy móc nhưng giảm bớt sai số tính toán.

(5) Viết giá trị phần nguyên từ đầu. → $(0.59375)_{10} = (0.10011)_2$


0.59375 × 2= 1 .1875 → (1) Viết phần thập phân xuống dưới.
0.1875 × 2= 0 .375 → (2) Viết phần thập phân xuống dưới.
0.375 × 2= 0 .75 → (3) Viết phần thập phân xuống dưới.
0.75 × 2= 1 .5 → (4) Viết phần thập phân xuống dưới.
0.5 × 2= 1 .0 ← Xử lý kết thúc khi phần thập phân bằng 0.

[5]

Để chuyển một số thập phân thành 1 số hệ 16, sử dụng 16 thay cho 2. Tổng quát, để chuyển 1 số thập phân thành 1 số hệ cơ số n, sử dụng *n* thay

[4] (Chú ý) Không có gì bảo đảm rằng nhân phần thập phân với 2 sẽ cho ra 0. Ta có thể kiểm tra đặc điểm này bằng ví dụ chuyển 0.110 thành số nhị phân, nó trở thành phân số nhị phân tuần hoàn. Luôn luôn có thể chuyển 1 phân số nhị phân thành 1 phân số thập phân, nhưng không có chiều ngược lại. Trong trường hợp đó, ta có thể dừng quá trình chuyển đổi ở một vị trí thích hợp.

[5] **Thập phân tuần hoàn**: một số thập phân có phần thập phân bị lặp vô hạn. Ví dụ 1/3 = 0.333..., và 1/7 = 0.142857142857..., có phần "3" và "142857" tương ứng lặp vô hạn.

cho 2.

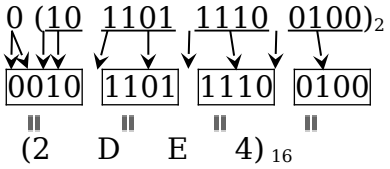
◆ Chuyển đổi giữa hệ 16 và hệ nhị phân

Sử dụng tính chất mỗi chữ số trong hệ 16 biểu diễn bằng 4 bit trong hệ nhị phân.

CHUYỂN TỪ HỆ NHỊ PHÂN SANG HỆ 16

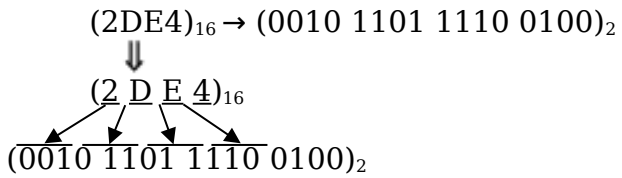
Xem ví dụ sau, chúng ta có thể nhóm số nhị phân thành từng nhóm 4 bit, bắt đầu từ bit nhỏ nhất (bit ngoài cùng bên phải), sau đó gán chữ số hệ 16 tương ứng cho từng nhóm. Nếu nhóm cuối cùng có ít hơn 4 bit, có thể thêm các chữ số 0 vào đầu.

$(10110111100100)_2 \rightarrow (10\ 1101\ 1110\ 0100)_2 \rightarrow (2DE4)_{16}$



CHUYỂN TỪ HỆ 16 SANG HỆ NHỊ PHÂN

Xem ví dụ dưới, ta có thể gán mỗi chữ số trong hệ 16 bởi số nhị phân 4-bit tương ứng.



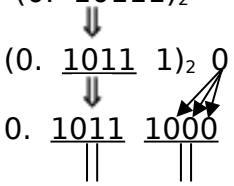
◆ Chuyển đổi phân số giữa hệ 16 và hệ thập phân

Để chuyển đổi giữa phân số hệ 16 và phân số hệ thập phân, ta có thể kết hợp phép chuyển đổi giữa hệ thập phân và hệ nhị phân với phép chuyển đổi giữa hệ nhị phân và hệ 16 để giảm lỗi.

CHUYỂN PHÂN SỐ THẬP PHÂN SANG HỆ 16

Ta có thể chuyển từ số thập phân sang số nhị phân trước, sau đó chuyển từ số nhị phân sang số hệ 16 tương ứng. Trong phép chuyển số nhị phân sang số hệ 16, ta có thể nhóm các bit thành từng nhóm 4-bit, bắt đầu từ bit lớn nhất (bit ngoài cùng bên trái) của phần phân số, và chuyển mỗi nhóm thành chữ số hệ 16 tương ứng. Nếu nhóm cuối cùng (ngoài cùng bên phải) có ít hơn 4 bit, có thể thêm các chữ số 0 vào cuối.

$(0.71875)_{10} \rightarrow (0.\ 10111)_2 \rightarrow (0.B8)_{16}$



$(0.\overline{B8})_{16} \Rightarrow 0.71875 = (0.B8)_{16}$

CHUYỂN TỪ HỆ THẬP PHÂN SANG HỆ 16[6]

Trước tiên, ta chuyển số hệ 16 thành số nhị phân tương ứng, sau đó chuyển số nhị phân sang số thập phân tương ứng.

$(0.B8)_{16} \rightarrow (0.10111000)_2 \rightarrow 0.71875$

$\Downarrow$

$(0.\underline{1011}\ \underline{1000})_2\ 0$

$\Downarrow$

$(0.10111000)_2 = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} = (0.71875)_{10}$

1.1.2 Biểu diễn số

Điểm chính	➢ Số thập phân được biểu diễn dạng gói đóng hoặc dạng vùng (gói mở) ➢ Số nhị phân được biểu diễn dạng dấu phẩy tĩnh hoặc dấu phẩy động.

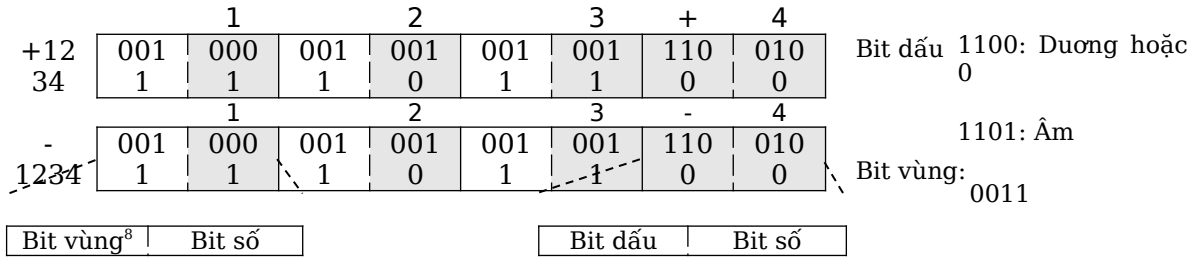
Số thập phân sử dụng hàng ngày cần được chuyển đổi sang một định dạng thuận tiện cho máy tính xử lý, có nhiều định dạng có thể biểu diễn giá trị số. Một vài định dạng biểu diễn các giá trị số trong máy tính được trình bày dưới đây.

Số thập phân	Thập phân dạng vùng	Tương thích cao với dữ liệu văn bản (cũng được biết tới như số thập phân gói mở)
	Thập phân gói đóng	Tốc độ xử lý nhanh hơn
Số nhị phân	Dấu phẩy tĩnh	Sử dụng cho dữ liệu số nguyên, ví dụ như chỉ số mảng…
	Dấu phẩy động	Sử dụng cho dữ liệu số thực như trong tính toán khoa học…

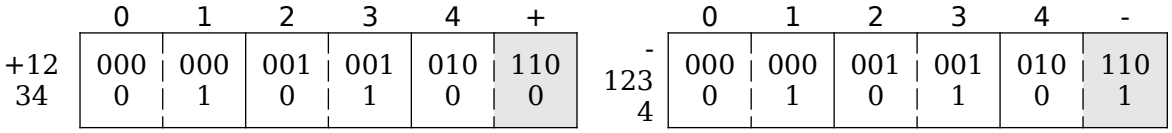
[6] (FAQ) Có nhiều câu hỏi trộn nhiều cơ số như "Đâu là đáp án đúng (trong dạng thập phân) của phép cộng các số hệ 16 và số nhị phân?" Nếu kết quả cuối cùng được biểu diễn trong dạng thập phân, tốt nhất chuyển các số nguyên bản thành số thập phân trước rồi tính. Nếu kết quả cuối cùng được biểu diễn trong cơ số khác 10 (nhị phân, hệ 8, hệ 16..), tốt hơn là chuyển số nguyên bản thành số nhị phân trước rồi tính.

◆ Biểu diễn số thập phân

Trong định dạng thập phân dạng vùng, mỗi chữ số của số thập phân được biểu diễn bằng 8 bit, và 4 bit cao nhất của chữ số cuối cùng được sử dụng để biểu diễn thông tin về dấu[7]. Các bit số của mỗi byte chứa giá trị số tương ứng trong hệ thập phân

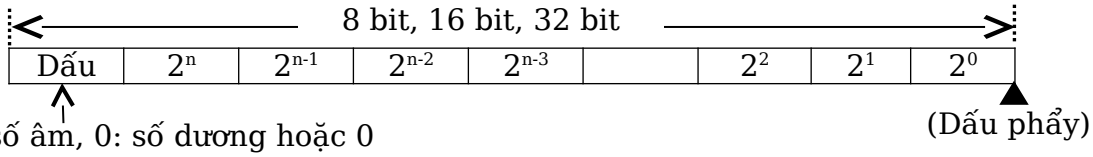


Trong định dạng thập phân gói đóng, mỗi chữ số của số thập phân được biểu diễn bằng 4 bit, và 4 bit cuối cùng xác định dấu. Khoảng trống phía đầu của byte cao nhất được thêm các bit 0. Mẫu bit dấu tương tự như trong định dạng thập phân mở gói. Trong ví dụ dưới đây, 2 bytes và 4 bit là đủ để biểu diễn số, nhưng cả hai trường hợp đều phải sử dụng 3 bytes bằng cách thêm 4 số 0 vào phần đầu tiên vì máy tính lưu trữ theo đơn vị byte[9].



◆ Biểu diễn số dấu phẩy tĩnh

Trong định dạng số dấu phẩy tĩnh, các số nhị phân nguyên được biểu diễn trong dạng số nhị phân có độ dài cố định. Phương pháp "bù 2" được sử dụng để biểu diễn số âm, với bit đầu tiên (bit dấu) của 1 số âm luôn là "1".



[7] (Gợi ý) Nếu dấu (dương hoặc âm) không được sử dụng trong định dạng thập phân dạng vùng, bit dấu được đặt giống với các bit phân vùng.
[8] (Chú ý) Mẫu bit trong các bit phân vùng là khác nhau tùy thuộc máy tính. Ví dụ dưới sử dụng "0011" nhưng một số máy sử dụng "1111". Các bit số là giống nhau.
[9] **Byte**: 1 byte là 1 đơn vị gồm 8 bit. Nó là đơn vị để biểu diễn các kí tự

Để biểu diễn số thập phân "-20" bằng phương pháp "bù 2", đầu tiên ta cần biểu diễn số thập phân "+20" trong dạng nhị phân:

$(+20)_{10}$ =	0	0	0	1	0	1	0	0	
				Đảo bit .					
	1	1	1	0	1	0	1	1	Phần bù 1[10]
+)								1	Cộng 1
$(-20)_{10}$ =	1	1	1	0	1	1	0	0	Phần bù 2

Vậy $(-20)_{10}$ được biểu diễn là $(11101100)_2$. Chiều dài bit là khác nhau giữa các hệ máy tính.

Tổng quát, các số từ -2^{n-1} tới $2^{n-1} - 1$, tổng cộng 2^n số, có thể được biểu diễn bằng n bit. Chú ý, chỉ xét giá trị tuyệt đối, 1 số âm có thể biểu diễn từ 1 số dương.

◆ Biểu diễn số dấu phẩy động

Trong số dấu phẩy động, 1 số thực được biểu diễn ở dạng mũ ($a = \pm m \times r^e$) sử dụng số nhị phân có chiều dài cố định, nên có thể biểu diễn được các số rất lớn hoặc rất nhỏ, vốn hay sử dụng trong tính toán khoa học. Tuy nhiên, do thanh ghi máy tính có số lượng bit là giới hạn, nên sai số có thể xảy ra trong biểu diễn giá trị thập phân tuần hoàn.

1 bit	8 bit		23 bit (độ chính xác đơn)
0	10000100		11010000000000000000000
↑	↑	▲	↑
Định trị dấu ±[11]	Số mũ e	Dấu phẩy	Định trị m

Đây là định dạng chuẩn quốc tế IEEE754

---

[10] **Phần bù (Complement)**: Phần bù của 1 số là giá trị nhận được bằng cách lấy 1 số cố định, là lũy thừa của cơ số hoặc lũy thừa của cơ số trừ 1, trừ đi số đó. Ví dụ, trong dạng thập phân, có phần bù 10 và phần bù 9. Trong hệ nhị phân có phần bù 2 và phần bù 1. Tổng quát trong hệ r bất kì, có phần bù r và phần bù (r – 1). Nếu x là 1 số n chữ số trong hệ cơ số r. Phần bù r của x là (r^n-x), và phần bù (r-1) của x là ((r^n-1)-x). Ví dụ, số 3 chữ số "123" trong hệ thập phân có phần bù sau: phần bù 10 là "1000 – 123 = 877," và phần bù 9 là "999 – 123 = 876." Số 4 bit "0101" trong hệ nhị phân có các phần bù sau: phần bù 2 là "10000 – 0101 = 1011 và phần bù 1 là "1111 – 0101 = 1010."

Bù của 10 **Bù của 9**

10^3=1000 10^3-1=999
-) 123 -) 123
877 876

Bù của 2 **Bù của 1**

2^4=10000 2^4-1=1111
-) 0101 -) 0101
1011 1010

Chú ý phần bù 1 trong hệ nhị phân chỉ là đảo của các bit (0 thành 1 và ngược lại). Phần bù 2 bằng phần bù 1 cộng 1.

[11] FAQ Có nhiều câu hỏi chuyển 1 số nhị phân cho trước thành số âm tương ứng và chuyển 1 số âm cho trước thành số dương tương ứng.

1.1.3 Biểu diễn dữ liệu không phải số

Điểm chính	➢ Mỗi kí tự được biểu diễn bởi 8 bit. ➢ Trong đa phương tiện, dữ liệu liên kết với dữ liệu ảnh tĩnh, dữ liệu ảnh động, và dữ liệu âm thanh được nắm giữ.

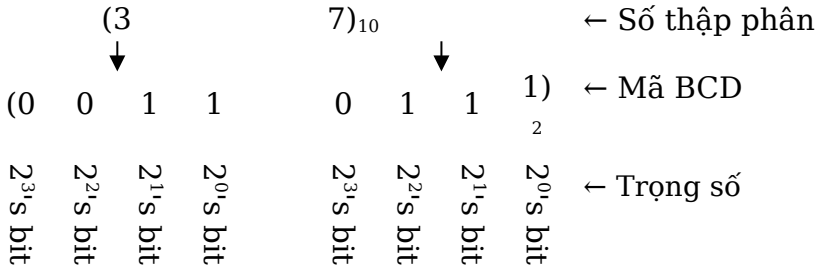
Biểu diễn phi số là biểu diễn của dữ liệu không phải các giá trị số. Nói cách khác, nó liên quan tới biểu diễn kí tự, âm thanh hoặc hình ảnh. Cách mà dữ liệu được biểu diễn là khác nhau giữa các máy tính, do đó để đảm bảo sự truyền dữ liệu giữa các máy tính suôn sẻ, cần thiết phải xây dựng các chuẩn biểu diễn dữ liệu.

◆ Biểu diễn kí tự

Với số nhị phân n-bit, có 2^n kiểu mã, tương ứng 1-1 của mã cho phép ta biểu diễn 2^n kí tự (chữ cái, chữ số, kí tự đặc biệt, và nhiều kí hiệu khác).

Mã BCD (Binary Coded Decimal Code)

Mỗi chữ số của số thập phân có thể biểu diễn bằng 4 bit. Xem ví dụ dưới.



Các bộ mã kí tự chuẩn

Mã	Giải thích
EBCDIC	Mã máy tính được định nghĩa bởi IBM cho các máy tính đa dụng. 8 bit biểu diễn 1 kí tự.
ASCII	Mã 7-bit được đưa ra bởi ANSI (American National Standards Institute – Viện tiêu chuẩn quốc gia Mĩ), sử dụng trên PC.
ISO code	ISO646 được đưa ra như 1 khuyến cáo của Tổ chức Tiêu Chuẩn Quốc Tế (ISO), dựa trên mã ASCII 7 bit, để trao đổi thông tin.
Unicode	Một chuẩn cho phép máy tính biểu diễn thống nhất các kí tự của tất hầu hết các nước. Mỗi kí tự dài 2 bytes.
EUC	Kí tự 2-byte và 1-byte có thể sử dụng đồng thời trên UNIX (mã UNIX mở rộng) Các ký tự tiếng Trung và Hàn cũng được xử lý.

◆ Biểu diễn hình ảnh và âm thanh

Lượng thông tin, như hình ảnh, âm thanh, kí tự được xử lý bởi hệ thống đa phương tiện là khổng lồ. Do đó kĩ thuật nén dữ liệu là yếu tố quyết định trong xây dựng 1 hệ thống đa phương tiện. Những công nghệ biểu diến cũng rất quan trọng. Mặt khác, dữ liệu đa phương tiện như ảnh tĩnh và âm thanh rất phổ biến trên PC khi kĩ thuật số hóa dữ liệu tương tự được cải tiến.

Ánh tĩnh	GIF	Định dạng để lưu trữ đồ họa, khả năng hiển thị 256 màu	
	JPEG[12]	Định dạng nén cho ảnh màu tĩnh, đưa ra bởi sự thống nhất của ISO và ITU-T.	
Ánh động	MPEG	Định dạng nén cho ảnh màu động, đưa ra bởi sự thống nhất của ISO và IEC.	
		MPEG-1	Dữ liệu được lưu trữ chính trên CD-ROM
		MPEG-2	Lưu trữ các hình ảnh video, ảnh thời gian thực
		MPEG-4	Chuẩn cho thiết bị đầu cuối di động
Âm thanh	PCM	Chuyển đổi tín hiệu tương tự (như âm thanh) thành tín hiệu số	
	MIDI	Giao diện kết nối một nhạc cụ với 1 máy tính	

1.1.4 Các phép toán và độ chính xác

Điểm chính

- Có 2 loại phép toán dịch: dịch số học và dịch logic.
- Các phép toán trong máy tính dựa trên số chữ số biểu diễn được, nên kết quả có thể xảy ra sai số tràn.

Máy tính được trang bị các mạch cho phép thực hiện 4 phép toán số học cơ bản và các phép dịch. Các phép toán như tính 2^n, tốc độ tính toán tăng lên do sử dụng phép dịch (hay di chuyển các chữ số). Tất cả các phép toán trên máy tính được thực hiện trên thanh ghi. Thanh ghi[13] này chỉ có hữu hạn chữ số có nghĩa, do đó kết quả 1 phép toán có thể chứa lỗi tràn.

[12] (FAQ) Có nhiều câu hỏi kiểm tra yêu cầu hiểu biết về tổ chức đã thiết lập các hàm và chuẩn về JPEG và MPEG. Một vài từ khóa như JPEG, ISO và ITU-T cho ảnh tĩnh, MPEG, ISO, IEC cho ảnh động cần được ưu tiên chọn trong bài làm.

[13] **Thanh ghi**: là bộ nhớ dung lượng thấp và tốc độ cao nằm trong CPU để chứa dữ liệu tạm thời, nó chứa các thanh ghi đa năng được sử dụng bởi CPU để thực hiện các thao tác.

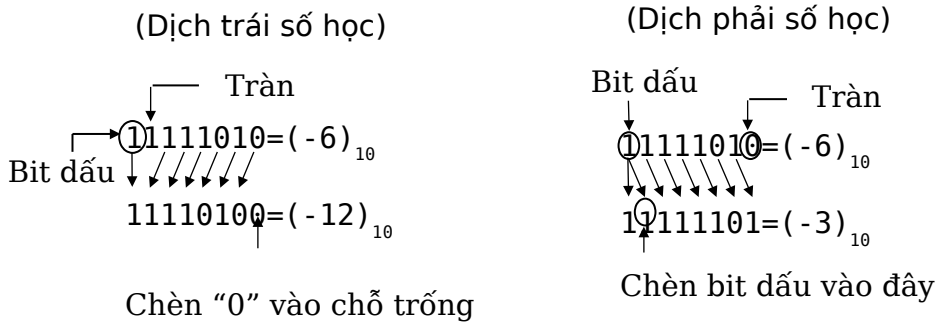
◆ Các phép dịch

Một **phép dịch** là 1 thao tác di chuyển chuỗi bit sang bên phải hoặc bên trái. Các phương pháp dịch được phân loại như bảng dưới.

	Dịch số học	Dịch logic
Dịch trái	Dịch số học trái	Dịch logic trái
Dịch phải	Dịch số học phải	Dịch logic phải

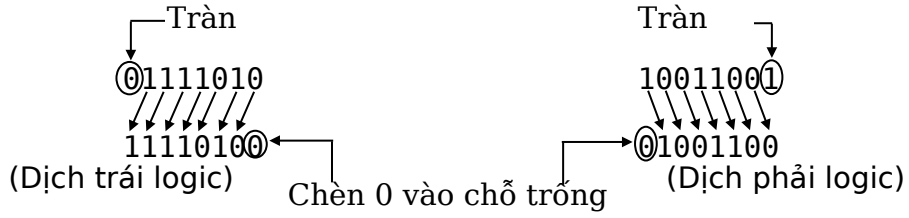
Dịch số học

Một phép dịch số học được sử dụng khi dữ liệu là dữ liệu số có dấu dương hoặc âm. Đó là 1 phép toán dịch chuỗi bit, trừ bit dấu, dùng cho số dạng dấu phẩy cố định. Phép **dịch số học sang trái** chèn 1 số "0" vào vị trí ngoài cùng bên phải (vị trí bị rỗng do dịch chuyển). Tổng quát, phép dịch số học sang trái n bit tăng số đó lên 2^n lần. Phép **dịch số học sang phải**, chèn bit dấu vào vị trí ngoài cùng bên trái (vị trí bị rỗng do dịch chuyển). Tổng quát, phép dịch số học sang phải n bit giảm số đó đi 2^{-n} lần ($1/2^n$). Ví dụ sau minh họa phép dịch số học. Dịch sang trái 1 bit nhân đôi giá trị trong khi dịch sang phải 1 bit giảm giá trị đi một nửa.



Dịch logic

Không giống như dịch số học, phép dịch logic không quan tâm dữ liệu là dạng số hay không, nó chỉ coi dữ liệu là một chuỗi bit. Nó dịch chuyển toàn bộ chuỗi bit dữ liệu và chèn 0 vào vị trí bị bỏ trống do dịch. Trong phép **dịch logic**[14], không có mối quan hệ như thay đổi giá trị 2^n hay 2^{-n} lần trong phép dịch số học. Ví dụ dưới đây minh học phép dịch logic 1 bit:



[14] (Chú ý) Trong 1 phép dịch logic, hình minh họa xác định rằng bit dấu 0 có thể trở thành 1 sau khi dịch. Nếu dữ liệu là số, nó nghĩa là 1 số dương chuyển thành 1 số âm bởi thao tác dịch.

◆ Sai Số

Khi các phép toán được thực hiện bởi các thanh ghi của máy tính với số chữ số giới hạn, các giá trị số không thể chứa trong thanh ghi sẽ bị bỏ qua, dẫn tới sự khác nhau giữa kết quả tính toán và kết quả thật. Sự khác nhau đó được gọi là **sai số**.

Sai số làm tròn

Do máy tính không thể xử lý số thập phân vô hạn, các bit nhỏ hơn bit xác định độ chính xác đều bị cắt bỏ, làm tròn lên hoặc làm tròn xuống tới giá trị giới hạn của số chữ số có nghĩa. Chênh lệch giữa giá trị thật và kết quả của phép làm tròn gọi là **sai số làm tròn**[15].

Mất chữ số có nghĩa

Khi một số trừ đi một số khác xấp xỉ nó hay khi 2 số, 1 số dương và 1 số âm có giá trị tuyệt đối xấp xỉ nhau cộng lại, số lượng chữ số có nghĩa có thể bị mất rất nhiều, đó là lỗi mất chữ số có nghĩa.

```
 )356.3622
-  356.3579
    0.0043
        ↑
        └── Khi kết quả gần bằng 0, số chữ số có nghĩa suy giảm trầm trọng.
```

Mất chữ số đuôi

Khi 1 số rất lớn cộng với 1 số rất nhỏ hoặc 1 số trừ đi 1 số khác, phần thông tin trong những bit thấp, không được chứa trong vùng biểu diễn, có thể bị mất do giới hạn độ dài của số. Hiện tượng này được gọi là **mất chữ số đuôi**. Để tránh sai số do mất chữ số đuôi, cần phải thực hiện các phép cộng, trừ theo thứ tự ưu tiên cho các số có trị tuyệt đối nhỏ.

```
  356.3622
-) 0.000015 ← Các chữ số có giá trị cực nhỏ có thể bị
  356.3622    bỏ qua.
```

[15] **Làm tròn**: Là cách xấp xỉ 1 số bằng cách làm tròn, làm tròn lên hoặc làm tròn xuống, hoặc làm tròn cho dễ hiểu với con người. Vì dụ nế 2.15 được làm tròn tới số nguyên gần nhất, nó được làm tròn thành 2 với sai số là 0.15.

Câu hỏi nhanh

Q1 Biểu diễn số thập phân 100 trong hệ nhị phân, hệ cơ số 8 và hệ cơ số 16.

Q2 Thực hiện phép dịch chuyển số học và dịch chuyển logic sang phải 3 bit trên số nhị phân 8 bit 11001100.

Q3 Giải thích khái niệm "triệt tiêu chữ số có nghĩa" và "mất chữ số đuôi"

A1 Trong hệ nhị phân: $(1100100)_2$
Trong hệ cơ số 8: $(144)_8$
Trong hệ 16: $(64)_{16}$
Chuyển sang hệ nhị phân:

$$100 = 64 + 32 + 4$$
$$= 2^6 + 2^5 + 2^2$$
$$= 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$
$$= (1100100)_2$$

Chuyển sang hệ 8: Chuyển sang hệ 16:

1 100 100 110 0100

↓ ↓ ↓ ↓ ↓

1 4 4_8 6 4_{16}

A2 Dịch số học: 11111001
Dịch logic: 00011001

(Dịch số học) (Dịch logic)

11001100 11001100

11111001 00011001

A3 Triệt tiêu chữ số có nghĩa: 1 hiện tượng xảy ra khi số chữ số có nghĩa giảm trầm trọng khi 1 số trừ đi 1 số khác gần bằng nó, hoặc khi cộng 1 số dương với 1 số âm có giá trị tuyệt đối gần bằng.

Mất chữ số đuôi: Hiện tượng khi phần thông tin trong những bit thấp, không được chứa trong vùng biểu diễn, có thể bị mất do giới hạn độ dài của số, khi 1 số rất lớn cộng với 1 số rất nhỏ hoặc 1 số trừ đi 1 số khác,

1.2 Thông tin và logic

Mở đầu

Để làm cho 1 máy tính thực hiện 1 nhiệm vụ, cần 1 chương trình được viết theo các luật. Chúng ta sẽ học về các phép toán logic, BNF và kí pháp Ba Lan ngược. Các phép toán logic là cơ sở cho phép toán cơ khí. BNF là các luật về cú pháp để viết chương trình. Kí pháp Ba Lan ngược được sử dụng để dịch các công thức toán học được viết trong chương trình.

1.2.1 Các phép toán logic

Điểm chính	➢ Các phép toán logic cơ bản: tổng logic, tích logic, phủ định logic, tổng loại trừ logic. ➢ Ngữ pháp của 1 chương trình được viết trong BNF.

Các phép toán logic cơ bản gồm: **tích logic (AND), tổng logic (OR), phủ định logic (NOT), và tổng loại trừ logic (EOR, XOR)**.

◆ Định nghĩa của các phép toán logic

Dưới đây là bảng kí hiệu và ý nghĩa các phép toán[16] logic đối với 2 biến logic A và B. Mỗi biến logic là số nhị phân 1 bit, nhận giá trị “1” hoặc “0”.

Phép toán logic	Kí hiệu	Ý nghĩa
Tích logic (AND)	$A \cdot B$	Kết quả là 1 khi cả 2 bit là 1.
Tổng logic (OR)	$A + B$	Kết quả là 1 khi ít nhất 1 bit là 1.
Phủ định logic (NOT)	$\overline{A}$	Đảo bit (0 thành 1, 1 thành 0)
Tổng loại trừ logic (EOR, XOR)	$A \oplus B$	Kết quả bằng 0 nếu 2 bit giống nhau và bằng 1 nếu 2 bit khác nhau

[16] (Chú ý) Trong máy tính được trang bị các mạch tương ứng với phép toán logic tích logic, tổng logic, phủ định logic. Tất cả các phép toán được thực hiện bằng các kết hợp các mạch này.

◆ Bảng chân lý / Các phép toán logic[17]

Bảng tổng hợp kết quả của các phép toán logic gọi là bảng chân lý[18]. Đây là bảng chân lý của các phép toán logic: tích logic, tổng logic, tổng loại trừ logic, phủ định logic.

A	B	Tích logic	Tổng logic	Tổng loại trừ logic	Phủ định logic	
		$A \cdot B$	$A + B$	$A \oplus B$	$\overline{A}$	$\overline{B}$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Tổng loại trừ logic có thể khai triển như sau. Nhiều câu hỏi có thể trả lời dễ dàng nếu biết dạng khai triển của tổng loại trừ logic, do đó cần nắm được công thức khai triển này.

$$A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$$

A	B	$\overline{A}$	$\overline{B}$	$A \oplus B$	$A \cdot \overline{B}$	$\overline{A} \cdot B$	$A \cdot \overline{B} + \overline{A} \cdot B$
0	0	1	1	0	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	0	0	0	0	0

◆ Định lý De Morgan's

Một tập các công thức nổi tiếng liên quan tới các phép toán logic là Định lý **De Morgan's.** Định lý xác định mối quan hệ như được chỉ ra dưới đây. Có thể dễ dàng ghi nhớ chúng nếu bạn nhớ sự hoán đổi giữa tích logic và tổng logic khi bỏ ngoặc. Nhiều câu hỏi có thể trả lời dễ dàng nếu biết định lý De Morgan's, nên hãy chắc rằng bạn đã nắm được các công thức.[19]

$$\overline{(A \cdot B)} = \overline{A} + \overline{B}$$

$$\overline{(A + B)} = \overline{A} \cdot \overline{B}$$

[17] **Phủ định của tổng logic:** $\overline{(A + B)} = \overline{A} \cdot \overline{B}$**. Phủ định của tích logic:** $\overline{(A \cdot B)} = \overline{A} + \overline{B}$

[18] (Chú ý) Một số bảng chân lý biểu diễn 1 bằng "T" (đúng) và 0 bằng "F" (sai)

[19] (FAQ) Nhiều câu hỏi có thể dễ dàng trả lời nếu biết định lý De Morgan. Có nhiều câu hỏi có thể dễ dàng trả lời nếu biết dạng khai triển của tổng loại trừ logic.

◆ Bộ cộng

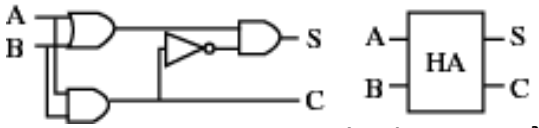
Một "bộ cộng" là một mạch thực hiện phép cộng số nhị phân 1 bit, bao gồm các mạch logic: AND, OR, NOT. Có 2 loại bộ cộng: bộ bán cộng, không đưa vào tổng số nhớ từ bit thấp hơn; bộ cộng đầy đủ, đưa vào tổng số nhớ từ các bit thấp hơn.

Bộ bán cộng

Khi một tín hiệu "0" hoặc "1" được gửi tới đầu vào A và B của mạch, kết quả phép cộng xuất hiện ở các đầu ra C và S. Ở đây C biểu thị cờ nhớ và S là bit thấp của kết quả phép cộng. Kết quả phép cộng được thể hiện dưới đây. Có thể thấy, C là tích logic và S là "tổng loại trừ logic"[20]

A		*B*		*C*	*S*
0	+	0	=	0	0
0	+	1	=	0	1
1	+	0	=	0	1
1	+	1	=	1	0

Trong hình vẽ dưới đây, cấu trúc mạch của bộ bán cộng được thể hiện ở bên trái. Hình bên phải là kí hiệu đơn giản của bộ bán cộng thường được sử dụng.



Kí hiệu đơn giản[21]

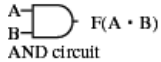
[20] (Gợi ý) Chắc chắn ràng bạn đã hiểu chính xác các phép toán nhị phân 1 bit. Thật cẩn thận vì rất dễ mắc phải lỗi. 4 phép cộng số 1 bit cho dưới đây.

A	0	0	1	1
B	+) 0	+) 1	+) 0	+) 1
	0	1	1	10

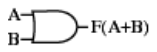
Nếu A và B đều là 1, phép cộng cho tổng là 2, nhưng trong hệ nhị phân, chỉ sử dụng 0 và 1, một cờ tràn được đặt, kết quả tổng là "10". Nếu mạch cộng không nhớ, đầu ra là "0".

[21] Bạn cần ghi nhớ kí hiệu mạch. Cẩn thận không lẫn lộn mạch AND, OR.

input output



AND circuit — Mạch AND



OR circuit — Mạch OR

NOT circuit — Mạch NOT

Bộ cộng đầy đủ

Một bộ cộng đầy đủ có 3 đầu vào, 1 trong số đó là cờ nhớ từ bit thấp hơn. Do đó, 1 bộ cộng đầy đủ cộng 3 giá trị X, Y, Z. Kết quả phép cộng được được thể hiện ở bảng dưới. Không như bộ bán cộng, không có mối quan hệ giữa tích logic và tổng loại trừ logic với bộ cộng đầy đủ.

X		*Y*	*Z*		*C*	*S*
0	+	0	0	=	0	0
0	+	0	1	=	0	1
0	+	1	0	=	0	1
0		1	1	=	1	0
1		0	0	=	0	1
1		0	1	=	1	0
1		1	0	=	1	0
1	+	1	1	=	1	1

Trong hình vẽ dưới, cấu trúc mạch của 1 bộ cộng đầy đủ được thể hiện ở bên trái. Từ hình vẽ thấy rằng bộ cộng đầy đủ gồm 2 bộ bán cộng kết hợp với nhau. Hình vẽ bên phải là kí hiệu đơn giản cho bộ cộng đầy đủ.

Kí hiệu đơn giản

1.2.2 BNF

Điểm chính	➢ Một cách biểu diễn chính xác văn phạm của 1 ngôn ngữ lập trình. ➢ Những kí hiệu kết thúc là không thể phân tích được nữa.

Để định nghĩa văn phạm của 1 ngôn ngữ lập trình (định nghĩa cú pháp) phải sử dụng các biểu thức không nhập nhằng. Để biểu diễn những văn phạm như vậy, dạng chuẩn BNF hay được sử dụng.[22]
BNF định nghĩa các luật của thứ tự của ký tự sử dụng chính các ký tự; nó cũng định nghĩa phép lặp và phép chọn sử dụng các ký hiệu đặc biệt. Vì chỉ có các ký tự được sử dụng trong định nghĩa nên biểu thức đơn giản và gần giống với cách biểu diễn câu. Hơn nữa, BNF không chỉ đưa ra các định nghĩa không nhập nhằng, nó còn khá dễ hiểu.

[22] (Chú ý) BNF được sử dụng lần đầu để định nghĩa ALGOL60, 1 ngôn ngữ lập trình cho tính toán kĩ thuật. BNF là 1 ngôn ngữ định nghĩa cú pháp chính thức, không quy định ngữ nghĩa. Do đó, nó không thể định nghĩa tất cả các luật của 1 ngôn ngữ, ngày nay nhiều phiên bản mở rộng của BNF được sử dụng.

◆ Dạng cơ bản của BNF

Biểu thức BNF chứa phép tuần tự, lặp và chọn.

Tuần tự

<x>::=<a><b>[23]

Kí hiệu đưa ra định nghĩa "phần tử cú pháp x là 1 chuỗi của các kí tự a và b". Ký hiệu "::=" có nghĩa "được định nghĩa là"

Lặp

<x>::=<a>...
Kí hiệu đưa ra định nghĩa "phần tử cú pháp x là 1 dãy lặp của kí tự a". Nó cũng có nghĩa là kí tự a được lặp 1 hoặc nhiều lần.

Chọn

<x>::=<a | b>

Kí hiệu đưa ra định nghĩa "phần tử cú pháp x nhận kí tự a hoặc kí tự b". Nếu 1 trong các lựa chọn bị bỏ qua, biểu thức tiếp theo được sử dụng:

<x>::=[<a>]

Kí hiệu đưa ra định nghĩa "phần tử cú pháp x nhận kí tự a hoặc kí tự rỗng". Kí hiệu "[ ]" có nghĩa có thể bỏ qua

◆ Kí hiệu kết thúc và không kết thúc

Một phần tử cú pháp đã được định nghĩa có thể dùng để định nghĩa phần tử khác hoặc chính nó. Những phần tử như vậy được gọi là những **"kí hiệu không kết thúc"** [24]. Những kí tự được sử dụng trực tiếp trong câu được gọi là "**kí hiệu kết thúc"**.

Trong định nghĩa sau, "<x>" là kí hiệu không kết thúc trong khi a, b và c là kí hiệu kết thúc.
<y>::=<a><x>
<x>::=<b><c>

[23] (Chú ý) **< >**: Ngoặc nhọn được sử dụng khi các kí tự được đặt liên tiếp hoặc khi đường biên không rõ ràng.
[24] (Chú ý) **Các kí hiệu không kết thúc**: được sử dụng làm cho định nghĩa cú pháp dễ hiểu.

◆ Ví dụ về BNF

Ví dụ, quy tắc cú pháp của "hằng dấu phẩy động" được định nghĩa như sau:

```
<hằng dấu phẩy động>::=      [<dấu>]<hằng cơ số>[<số mũ>] |
                             [<dấu>]<chuỗi số><số mũ>
<hằng cơ số>  ::= [<chuỗi số>]<.><chuỗi số>|<chuỗi số><.>
<số mũ>       ::= <E>[<dấu>]<chuỗi số>
<chuỗi số>    ::= <chữ số>|<chuỗi số><chữ số>
<chữ số>      ::= 0|1|2|3|4|5|6|7|8|9
<dấu>         ::= +| –
```

Hãy dựa vào các quy tắc cú pháp trên để xem <hằng dấu phẩy động> trông có rõ ràng[25].

Để giải thích, ta viết các phần tử như sau:

<hằng dấu phẩy động>::=[<dấu>]<hằng cơ số>[<số mũ>] | [<dấu>]<chuỗi số><số mũ>

(1) (2) (3) (4) (5) (6)

Định nghĩa <hằng dấu phẩy động> được phân tách bởi đường "|", chúng phân tách nhóm (1)~(3) với nhóm (4)~(6), nên có thể có 2 dạng. Ta lấy nhóm thứ nhất (1)~(3) làm ví dụ để xem đâu là <hằng dấu phẩy động>. Mỗi phần tử (1)~(3) có thể trải ra như sau:

<dấu><hằng cơ số><số mũ>

<E><dấu><chuỗi số>

<chuỗi số><.><chuỗi số>

Ở đây, cần phân tích <chuỗi số>. [26]

<chuỗi số> được định nghĩa như sau:

<chuỗi số>::= <chữ số>|<chuỗi số><chữ số>

Tiếp theo, cần dịch <chữ số>, nó được định nghĩa như sau:

<chữ số>::= <0|1|2|3|4|5|6|7|8|9>

Nghĩa là, biểu thức dưới đây là đúng:

<chữ số>::= 0

[25] (FAQ) Một ví dụ về quy tắc cú pháp: thường, các câu hỏi lựa chọn các câu thỏa mãn các quy tắc cú pháp được đưa ra

[26] (Chú ý) Một biểu diễn với 1 chuỗi kí tự kết hợp với các kí hiệu đặc biệt (như $, *) được gọi là biểu thức chính quy. Các kí tự xác định được gọi là các siêu kí tự. Các siêu kí tự có ý nghĩa đặc biệt. Trong UNIX, Windows...nếu tìm kiếm 1 file bằng cách nhập "*.jpg" thì hệ thống sẽ tìm tất cả các file có mở rộng là "jpg". Ở đây, kí hiệu "*" là 1 siêu kí tự.

Hơn nữa, từ định nghĩa <chuỗi số> và <chữ số>, “0” cũng là <chuỗi số>. Do đó, ta có biểu diễn sau:

<chuỗi số>::= 0<chữ số>

Vậu “01” cũng là < chuỗi số>. Tiếp tục.

<chuỗi số>::= 01<chữ số> ,

“012” cũng là <chuỗi số>. Do đó, <chuỗi số> là bất cứ chuỗi liên tiếp các chữ số. Vậy, ví dụ <hằng cơ số> trông như sau:

<hằng cơ số>::= 123.456

Nên, phân tích của <hằng dấu phẩy động> có thể đưa cho ta ví dụ sau:

<dấu><hằng cơ số><số mũ>
 +   123.456   E+123

Dĩ nhiên, <dấu> có thể là dấu âm “–” hoặc có thể bỏ qua, chuỗi sau có thể là hằng số dấu phẩy động.

-123.456E-123
-123.456E123

Như thấy ở trên, nếu tất cả các dạng đặc biệt đều được biểu diễn, cho kết quả là một lượng thông tin khổng lồ. BNF được sử dụng để đưa ra định nghĩa chung nhằm tránh tình huống này.

1.2.3 Kí pháp Ba Lan ngược

Điểm chính	➢ Kí pháp Ba Lan ngược là cách để dịch các biểu thức toán học. ➢ Nó đặc trưng bởi 2 biến theo sau bởi 1 toán tử.

Kí pháp Ba Lan ngược là 1 phương pháp biểu diễn các công thức toán học được sử dụng hàng ngày sang dạng biểu diễn dễ xử lý hơn bởi máy tính. Khái niệm cơ bản của kí pháp này là toán tử được đặt vào cuối thay vì đặt vào giữa của công thức
Ví dụ, $X = A + B * C$ nghĩa là "tính tích của B và C, cộng A, sau đó đưa kết quả tới X", được biểu diễn như sau:

$XABC*+=$

◆ Chuyển đổi công thức toán học thành ký pháp Ba Lan ngược

Ví dụ, chuyển "$e=a-b\div(c+d)$" sang **kí pháp Ba Lan ngược**[27]**.** Thứ tự của các phép toán là thứ tự thường dùng. Phần gạch chân được tính trước[28]:

(1) $e=a-b\div(\underline{c+d})$
"(c+d)" được chuyển sang kí pháp Ba lan ngược → $cd+$
Ta gọi nó là chuỗi "P"

(2) $e=a-\underline{b\div P}$
"$b\div P$"được chuyển sang kí pháp Ba lan ngược → $bP\div$
Ta gọi nó là chuỗi "Q."

(3) $e=\underline{a-Q}$

"a – Q" chuyển sang kí pháp Ba lan ngược → $aQ-$
Ta gọi nó là chuỗi "R"

(4) $\underline{e=R}$
"e = R" được chuyển sang kí pháp Ba lan ngược → $eR=$

[27] (FAQ) Chuyển sang kí pháp Ba Lan ngược hoặc chuyển sang công thức toán học là các chủ đề hay gặp trong bài thi. Tốt nhất là học cách trả lời các câu hỏi này bằng trực quan.
[28] (Chú ý) Bằng trực quan, kí pháp Ba Lan ngược có thứ tự các phép toán trong công thức khi chuyển đổi như sau.

e=a-b/(c+d)
①cd+
②b①/
→bcd+/
③a②-
→abcd+/-
④e③=
→eabcd+/-=

(5) Viết lại P, Q, R trong dạng kí pháp Ba Lan ngược (gạch chân xác định nơi xảy ra thay thế):

$eR= \rightarrow \underline{eaQ-} = \rightarrow \underline{eabP\div-} = \rightarrow \underline{eabcd+\div-} =$

◆ Chuyển từ kí pháp Ba Lan ngược thành công thức toán học

Cách chuyển từ kí pháp Ba Lan ngược thành công thức toán học như sau:

(i) Quét kí pháp Ba Lan ngược từ đầu, tìm 1 toán tử.[29]
(ii) Thực hiện phép toán xác định bởi toán tử đầu tiên, sử dụng 2 biến ngay liền trước toán tử.
(iii) Đặt kết quả của phép toán ở (ii) là 1 biến mới và lặp lại 2 bước đầu tiên (i) và (ii).

Ví dụ, cho kí pháp Ba Lan ngược sau "$eabcd+\div-=$." Nó được chuyển như sau:

Ở đây, phần gạch chân xác định phần được chuyển đổi

(1) Quét "$\underline{eabcd+\div-} =$" từ đầu, tìm kiếm 1 toán tử. Toán tử đầu tiên là "+", chú ý tới toán tử đó và 2 biến liền trước nó, tức là "$cd+$."

$cd+ \rightarrow c+d$

Đặt nó là *P*. → "$eabP\div-=$"

(2) Quét biểu thức "$\underline{eabP\div-} =$" từ đầu để tìm kiếm 1 toán tử. Toán tử đầu tiên là "÷," chú ý tới toán tử đó và 2 biến liền trước nó, tức là "$bP\div$."

$bP\div \rightarrow b\div P$

Đặt nó là *Q*. → "$eaQ-=$"

(3) Quét biểu thức "$\underline{eaQ-} =$" từ đầu, tìm kiếm 1 toán tử. Toán tử đầu tiên là "–," nên tập trung vào toán tử đó và 2 biến liền trước nó, tức là "$aQ-$."

$aQ- \rightarrow a-Q$

Đặt nó là *R*. → "$eR=$"

(4) Viết lại *P*, *Q*, và *R* như công thức toán học (phần gạch chân được thay thế)

$eR= \rightarrow \underline{e=R} \rightarrow \underline{e=a-Q} \rightarrow \underline{e=a-b\div P} \rightarrow e=a-(b\div(c+d))$

Bỏ ngoặc không cần thiết, ta được kết quả sau:

$e=a-b\div(c+d)$

◆ Kí pháp Ba Lan

Trong **kí pháp Ba Lan**, "$a+b$" được biểu diễn là "$+ab$"[30]. Trong khi biểu diễn trong kí pháp Ba Lan ngược là "$ab+$," Kí pháp Ba Lan đặt các toán tử trước các biến. Các khái niệm cơ bản tương tự như trong kí pháp Ba Lan

[29] (Gợi ý) trong kí pháp Ba Lan ngược, khi tìm kiếm 1 toán tử, ta sẽ luôn có 2 biến nằm ngay trước nó.
[30] Trong kí pháp Ba Lan, mọi toán tử nằm trước 2 biến. Nếu không có 2 biến, hãy tìm kiếm biến tiếp theo.

ngược. Nếu chuyển "$e = a - b \div (c + d)$" thành dạng kí pháp Ba Lan ta có:

$e = a - b \div (c + d)$ → $= e - a \div b + cd$

Câu hỏi nhanh

Q1 Cho giá trị của các biến logic x và y như sau. Hoàn thành bảng dưới bằng cách tính tích logic, tổng logic và tổng loại trừ logic.

x	*y*	Tích logic	Tổng logic	Tổng loại trừ logic
0	0			
0	1			
1	0			
1	1			

Q2 Giải thích khái niệm "bộ cộng", "bộ bán cộng" và "bộ cộng đầy đủ".

Q3 Chuyển công thức "$(a + b) \times (c - d)$" sang dạng kí pháp Ba Lan ngược.

A1

x	*y*	Tích logic	Tổng logic	Tổng loại trừ logic
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

A2

Bộ cộng: Mạch cộng các số nhị phân 1 bit. Gồm các mạch AND, OR, NOT.

Bộ bán cộng: Một bộ cộng mà không đưa vào kết quả số nhớ từ các bit thấp hơn. Nó có 2 đầu vào và 2 đầu ra.

Bộ cộng đầy đủ: Một bộ cộng mà đưa vào kết quả số nhớ từ các bit thấp hơn. Nó có 3 đầu vào và 2 đầu ra.

A3

$ab+cd-\times$

Dịch là "cộng a và b", "trừ c cho d" và "nhân" các kết quả với nhau.

1.3 Cấu trúc dữ liệu

Mở đầu

Khi tìm kiếm giải thuật cho một chương trình, sẽ dễ dàng hơn để tạo ra một giải thuật nếu đặt dữ liệu trong các mẫu tiêu biểu. Các mẫu đặc trưng được gọi là các cấu trúc dữ liệu. Một số cấu trúc dữ liệu quen thuộc là: mảng, danh sách, ngăn xếp, hàng đợi và cây.

1.3.1 Mảng (Array)

Điểm chính	➢ Mảng có thể sử dụng trong mọi cấu trúc dữ liệu ➢ Mảng được truy cập qua chỉ số.

Mảng là một cấu trúc dữ liệu tạo bởi nhiều dữ liệu cùng kiểu. Ví dụ, hình ảnh lũ trẻ xếp thành một hàng đơn. Trong tình huống này, các đối tượng có các thuộc tính giống nhau (ở đây đối tượng là "trẻ con") được lặp lại, tương tự như một mảng. Mỗi đứa trẻ được xác định như "đứa thứ nhất, đứa thứ hai…". Các số "1, 2, …" được gọi là các chỉ số. Một mảng được sử dụng khi nhiều dữ liệu cùng kiểu được quản lý không phải riêng lẻ mà trong mối quan hệ với nhau. Dữ liệu được truyền qua tên mảng, mỗi trường dữ liệu (phần tử) được xác định bởi một chỉ số.

◆ Mảng một chiều

"Mảng một chiều" là khái niệm như sau[31]:

Chỉ số	1	2	3	4	…	25	26
Mảng *T*	*a*	*b*	*c*	*d*	*…*	*y*	*z*

Mỗi mảng có một tên. Trong ví dụ trên, tên mảng là "T". Để xác định mỗi phần tử sử dụng chỉ số. Chỉ số chỉ ra vị trí của phân tử trong mảng[32]. Ví dụ, phần tử thứ tư "*d*" được xác định bởi "T(4)", chỉ số nằm trong ngoặc đơn. Một vài ngôn ngữ sử dụng ngoặc vuông [ ]. Tổng quát, phần tử thứ n của mảng được biểu thị bằng "T(n)". Bằng cách thay đổi giá trị của n, chúng ta có thể xác định bất cứ phần tử nào của mảng.

[31] (FAQ) Có những câu hỏi khó về chính các mảng. Vì, các câu hỏi về cấu trúc dữ liệu hoặc giải thuật luôn sử dụng mảng. Do đó cần hiểu rõ tính chất của mảng. Đặc biệt, cần phải hiểu rõ cách sử dụng chỉ số.

[32] (Gợi ý) Chỉ số bắt đầu từ 0 trong một số ngôn ngữ lập trình. Các câu hỏi về giải thuật trong bài thi có thể đánh chỉ số bắt đầu từ 0 hoặc 1, hãy cẩn thận.

◆ Mảng hai chiều

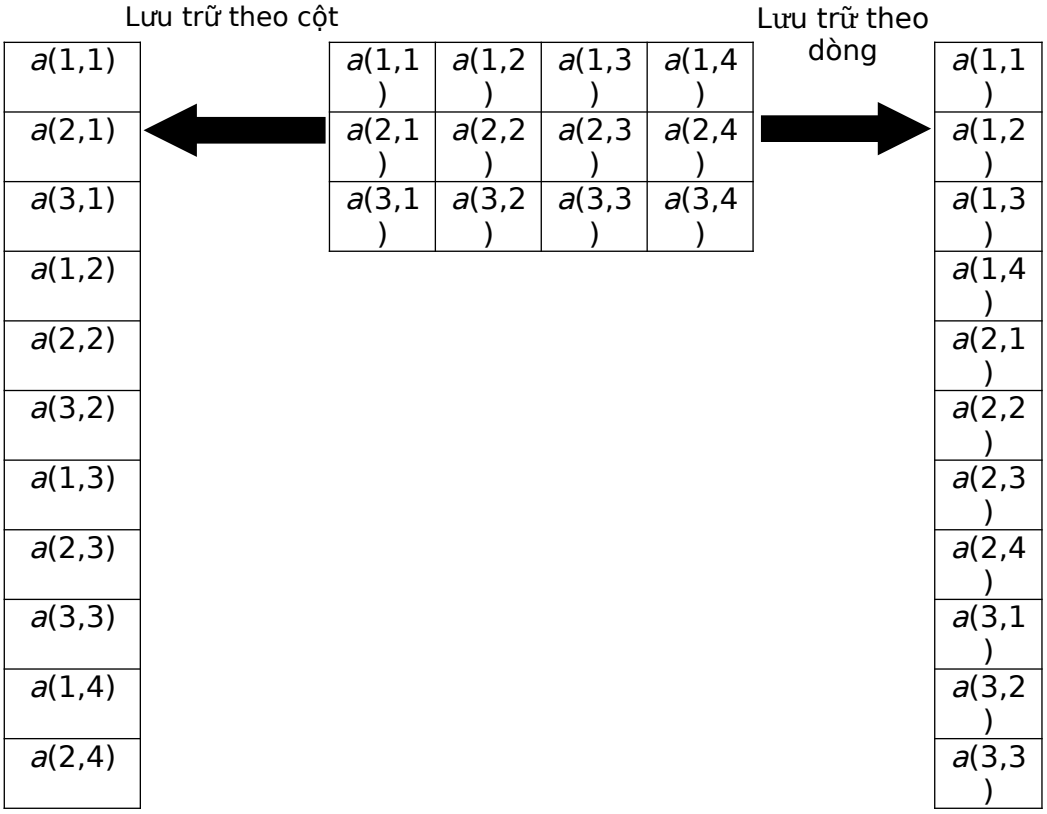
Một **mảng hai chiều** là khái niệm như sau[33]:

	Cột 1	Cột 2	Cột 3	Cột 4
Dòng 1	*a* (1,1)	*a* (1,2)	*a* (1,3)	*a* (1,4)
Dòng 2	*a* (2,1)	*a* (2,2)	*a* (2,3)	*a* (2,4)
Dòng 3	*a* (3,1)	*a* (3,2)	*a* (3,3)	*a* (3,4)

Tổng quát, phần tử của mảng hai chiều được xác định bằng bộ 2 chỉ số *m* và *n*. Kí hiệu là "*a*(*m*,*n*)" hoặc "a_{mn}" với *m* là chỉ số dòng và *n* là chỉ số cột. Mảng trên là mảng hai chiều với 3 dòng và 4 cột, còn được gọi là "mảng 3 nhân 4".

◆ Lưu trữ theo dòng và lưu trữ theo cột

Khi một mảng được lưu trữ trong bộ nhớ, nó được lưu trữ như mảng một chiều. Khi các phần tử của một mảng hai chiều được lưu trữ như mảng một chiều, có thể sử dụng 2 phương pháp, dựa trên thứ thự các phần tử được lưu trữ: **lưu trữ theo dòng** hoặc **lưu trữ theo cột**[34]**.**



[33] (Gợi ý) Mảng 1 chiều được sử dụng khi dữ liệu được lưu trữ đơn giản. Một mảng 2 chiều được sử dụng khi các đối tượng lưu trữ giống các ma trận toán học.
[34] Giữa các ngôn ngữ lập trình, Fortran sử dụng lưu trữ theo cột trong khi COBOL, PL/I và C sử dụng lưu trữ theo dòng.

Trong hình trên, chú ý sự khác nhau của chỉ số. Trong phương pháp lưu trữ theo cột, chỉ số *x* trong *a(x, y)* thay đổi trước. Trong phương pháp lưu trữ theo dòng, chỉ số *y* thay đổi trước. Khi nói tới mảng, tìm kiếm các phần tử tuần tự sẽ hiệu quả hơn truy cập mà bỏ qua chỗ này chỗ khác. Do vậy. để xử lý hiệu quả, chỉ số được điều khiển như sau:

- Theo cột: *x* trong "*a*(*x*,*y*)" (*x* = 1 tới m; *y* = 1 tới *n*) thay đổi trước.
- Theo dòng: *y* trong "*a*(*x*,*y*)" (*x* = 1 tới *m*; *y* = 1 tới *n*) thay đổi trước.

Với sự sắp xếp này, tham chiếu tới 1 mảng hai chiều được làm cho hiệu quả hơn khi chuyển nó thành 1 mảng một chiều.

1.3.2 Danh sách (List)

Điểm chính	➢ Danh sách được đặc trưng bởi các con trỏ được liên kết ➢ Các thao tác trên danh sách được điều khiển bởi sự thay đổi giá trị của các con trỏ.

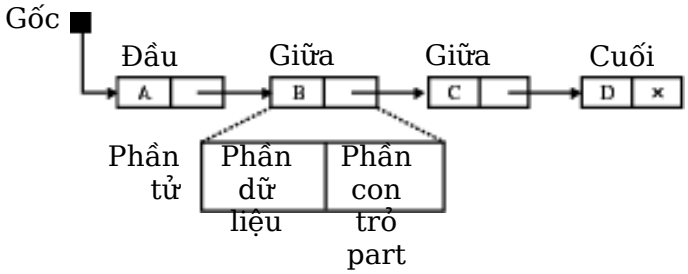
Danh sách là một tập của các dữ liệu giống nhau hoặc tương tự nhau được đặt một cách logic trên một hàng (tuyến tính[35]); cấu trúc của nó gần giống 1 mảng. Điểm khác biệt là ở chỗ, trong khi các phần tử của mảng được đặt kề nhau một cách vật lý, các phần tử của danh sách có thể đặt ở những vị trí độc lập và có các con trỏ liên kết giữa chúng. Vì vậy, mảng và danh sách chỉ khác biệt ở điểm: mảng nói tới một danh sách tuyến tính, và danh sách nói tới một danh sách móc nối (do các phần tử được móc nối bởi con trỏ).

Tổng quát, thuật ngữ "danh sách" nói tới "danh sách móc nối". Do đó chúng ta gọi "danh sách móc nối" đơn giản là "danh sách"

◆ Cấu trúc của danh sách

Một danh sách là một cấu trúc dữ liệu mà các phần tử được kết nối bởi các con trỏ. Một con trỏ là thông tin xác định vị trí lưu trữ (địa chỉ) của phần tử kế tiếp. Mỗi phần tử được kết nối bởi 1 con trỏ, nên các phần tử không cần được đặt theo thứ tự.

Một danh sách có thể có cấu trúc đa dạng. Hình vẽ sau được gọi là một danh sách 1 chiều.[36]



Con trỏ trỏ tới phần tử đầu tiên được lưu trữ trong một biến gọi là gốc. Phần tử cuối cùng của danh sách không có phần tử nào sau nó, nên con trỏ của nó chứa ký hiệu (X) xác định rằng đó là phần tử cuối cùng của danh sách. Trong vài ngôn ngữ lập trình, kí hiệu này có thể được lưu trữ tự động, một số khác, có thể là bất cứ ký hiệu gì. Điều quan trọng là nó để gán cho một giá trị không tồn tại như dữ liệu

[35] (Gợi ý) Thuật ngữ "tuyến tính" chỉ tới 1 tập dữ liệu nằm liền kề nhau. 1 mảng là tuyến tính khi các phần tử được đặt trong vùng liên tục. Nói cách khác, một danh sách là 1 cấu trúc mà các phần tử được liên kết bởi con trỏ, chúng có thể không nằm trong vùng liên tục.

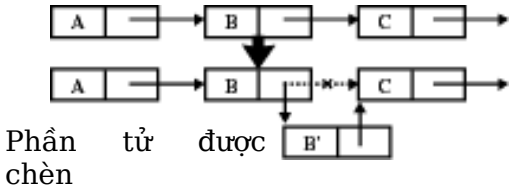
[36] Bên cạch danh sách 1 chiều còn có danh sách 2 chiều và danh sách vòng. 1 danh sách 2 chiều là danh sách trong đó mỗi phần tử có 1 con trỏ xác định phần tử liền trước và 1 con trỏ xác định phần tử liền sau. Một danh sách vòng là một danh sách mà phần tử cuối cùng có con trỏ xác định vị trí của phần tử đầu tiên.

Các thao tác cơ bản với danh sách

Có một số thao tác cơ bản được thực hiện trên danh sách, trong đó thao tác quan trọng nhất là chèn và xóa danh sách.

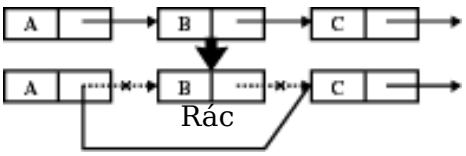
Chèn[37]

Để chèn thêm một phần tử vào một danh sách cho trước. Tất cả những thứ cần làm là thay đổi một số con trỏ thích hợp. Đầu tiên, con trỏ của phần tử được chèn nhận địa chỉ của phần tử mà sẽ nằm ngay sau nó trong danh sách. Tiếp theo, thay đổi con trỏ của phần tử ngay trước phần tử được chèn vào để nó có địa chỉ của phần tử được chèn vào



Xóa

Để xóa một phần tử khỏi một danh sách cho trước, giống như chèn thêm vào, ta cần thay đổi các con trỏ. Thay đổi con trỏ của phần tử ngay phía trước phần tử bị xóa để nó chỉ tới phần tử ngay sau phần tử bị xóa. Dữ liệu bị xóa tồn tại như rác đến khi danh sách được tái cấu trúc, do dó vào những thời điểm phù hợp cần tiến hành **dọn rác**[38] để xóa các phần tử không cần thiết[39].



[37] (FAQ) Nhiều câu hỏi liên quan tới thao tác chèn vào và xóa khỏi danh sách. Cần phải xem xét cẩn thận phần tử mà con trỏ của nó được lưu trữ.

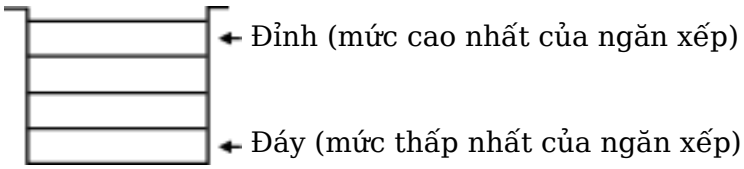
[38] **Dọn rác**: Là một thủ tục nhờ đó bộ nhớ nhỏ, phân mảnh, không sử dụng và các vùng không sử dụng khác do rò rỉ bộ nhớ được kết hợp với nhau để tăng vùng nhớ có thể sử dụng. Nếu rác không được dọn, vùng nhớ có thể sử dụng sẽ tiếp tục giảm và cuối cùng chức năng khởi động lại hệ thống sẽ được gọi.

[39] **Rò rỉ bộ nhớ**: Nó chỉ tình huống mà bộ nhớ chính được tìm tự động bởi 1 ứng dụng không được giải phóng vì vài lí do và nằm lại trong bộ nhớ chính. Để khử rò rỉ bộ nhớ, dọn rác là cần thiết.

1.3.3 Ngăn xếp (Stack)

Điểm chính	➢ Ngăn xếp là cấu trúc dữ liệu kiểu LIFO (vào cuối ra đầu/ vào sau ra trước). ➢ Ngăn xếp được sử dụng để quản lý địa chỉ trở về của chương trình con

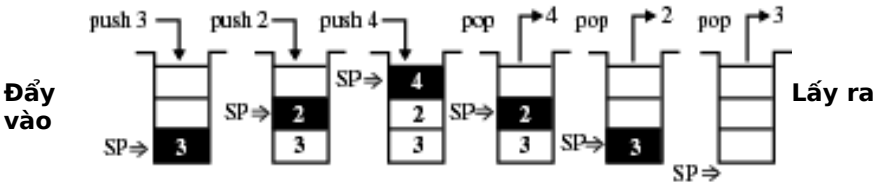
Ngăn xếp là một cấu trúc dữ liệu trong đó dữ liệu chèn thêm vào và xóa đi đều nằm ở cùng một đầu của danh sách. Nó được minh họa như hình dưới.



Điểm đầu nơi các phần tử thêm vào và xóa được đi gọi là "đỉnh" và đầu kia gọi là "đáy". Thao tác thêm vào được gọi là "đẩy xuống" và xóa đi gọi là "lấy ra".

◆ Các thao tác cơ bản với ngăn xếp

Ngăn xếp là một cấu trúc dữ liệu kiểu **LIFO** (Last-In First-Out), nghĩa là phần tử được lưu trữ cuối cùng sẽ được lấy ra đầu tiên. Trong hình vẽ dưới, dữ liệu được lưu trữ theo thứ tự "3→2→4" và được lấy ra theo thứ tự "4→2→3."[40]



Một con trỏ gọi là "Con trỏ ngăn xếp" (Stack Point-SP) được sử dụng để giữ vết đỉnh của ngăn xếp hiện tại, chúng ta có thể chèn thêm một phần tử hoặc xóa một phần tử khỏi vị trí được xác định bởi SP. Con trỏ ngăn xếp có thể trỏ tới đỉnh thật của ngăn xếp, thỉnh thoảng trỏ tới một điểm khác tùy vào cài đặt.

◆ Sử dụng ngăn xếp

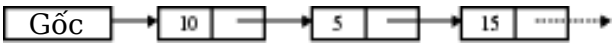
Khi một chương trình chính gọi một chương trình con (thủ tục con) hoặc một hàm, địa chỉ trở về của chương trình đang được thực hiện được lưu trữ trong ngăn xếp, khi chương trình con hoàn thành, địa chỉ trở về của chương trình chính được lấy ra khỏi ngăn xếp và trả về cho trình điều khiển. Hơn nữa, nếu một chương trình con gọi một chương trình con khác,

[40] (FAQ) Có nhiều câu hỏi liên quan tới ngăn xếp. Mẫu câu được sử dụng thường xuyên là các câu hỏi về điều gì sẽ xảy ra với ngăn xếp đã cho khi lặp lại thao tác đẩy vào, lấy ra.

địa chỉ trở về của chương trình gọi được lưu trữ tuần tự trong ngăn xếp.[41]

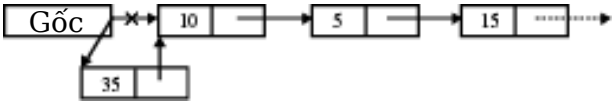
◆ Cài đặt ngăn xếp sử dụng danh sách

Sử dụng cấu trúc của một danh sách, ta có thể cài đặt một ngăn xếp. Trong trường hợp của danh sách sau, một phần tử chỉ có thể thêm vào hoặc xóa đi từ đỉnh của danh sách.



Chèn vào danh sách

Để thêm phần tử "35" vào danh sách, ta đặt nó làm phần từ đầu tiên, tức là trước phần tử "10".



Xóa khỏi danh sách

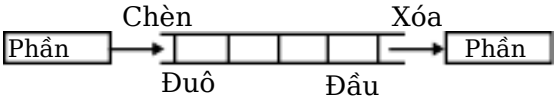
Ta xóa phần tử đầu tiên "35" khỏi danh sách, nơi nó được chèn vào ở xử lý trên. Hệ quả là, bằng cách kết hợp thao tác chèn và xóa, ta có thể cài đặt ngăn xếp.

1.3.4 Hàng đợi (Queue - Danh sách đợi)

Điểm chính	➢ Hàng đợi là một cấu trúc dữ liệu kiểu FIFO (First-In First-Out) vào trước ra trước. ➢ Hàng đợi được sử dụng trong xử lý giao dịch trực tuyến.

Hàng đợi là một cấu trúc dữ liệu mà thao tác chèn được thực hiện ở một đầu và thao tác xóa (lấy ra) được thực hiện ở đầu kia. Dựa trên khái niệm, nó có được mô tả như sau.

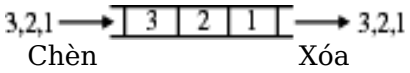
[41] Sử dụng 1 ngăn xếp, 1 chương trình con có thể được gọi từ 1 chương trình con khác. Mỗi lần 1 chương trình con được gọi liên tiếp, địa chỉ trở về được lưu trữ trong ngăn xếp. Khi lấy ra theo thứ tự ngược với thứ thự lưu trữ, chương trình con được trở về theo thứ tự ngược lại. 1 cấu trúc mà 1 chương trình con được gọi từ một chương trình con khác là cấu trúc lồng.



Dữ liệu đầu tiên trong hàng đợi được gọi là **"đầu"** và dữ liệu cuối cùng được gọi là **"đuôi".** Một hàng đợi còn được gọi là "danh sách đợi – waiting list", tên này lấy từ khái niệm của trình tự xử lý.

◆ Các thao tác trên hàng đợi

Hàng đợi là cấu trúc dữ liệu kiểu FIFO (First-In First-Out) vào trước ra trước, nghĩa là phần tử được lưu trữ trước sẽ được lấy ra trước. Trong hình vẽ dưới, dữ liệu được lưu trữ theo thứ tự "1 → 2 → 3" và được lấy ra theo thứ tự "1 → 2 → 3."



Trong hàng đợi, dữ liệu luôn luôn được lưu trữ (enqueued) sau dữ liệu cuối cùng, và dữ liệu đầu tiên (cũ nhất) luôn luôn bị xóa (dequeued) trước.[42]

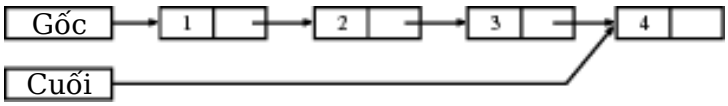
◆ Các ví dụ về hàng đợi

Trong xử lý đa chương trình, các chương trình đang đợi để thực hiện có độ ưu tiên bằng nhau được đặt vào hàng đợi thực hiện và chờ đến khi CPU sẵn sàng. Trong **xử lý các giao dịch trực tuyến[43],** các thông điệp (văn bản điện tử) được đưa vào hàng đợi và được xử lý theo thứ tự đi vào.

◆ Cài đặt hàng đợi sử dụng danh sách

Để cài đặt hàng đợi từ danh sách, cần tìm con trỏ xác định vị trí của phần tử cuối cùng của danh sách. Thao tác chèn được thực hiện ở điểm cuối của danh sách và thao tác xóa được thực hiện ở đầu[44].

Cho danh sách như sau, ở đây con trỏ tới phần tử cuối cùng được gán cho "đuôi" để thuận tiện.



Vì ta giả sử rằng phần tử được chèn vào cuối của danh sách, hình vẽ trên xác định rằng phần tử được chèn vào và lưu trữ theo thứ tự "1 → 2 → 3 → 4."

[42] (Chú ý) Các ví dụ về hàng đợi được thấy quanh ta hàng ngày. Ví dụ, 1 hàng người đợi mua vé tàu từ 1 máy bán vé là 1 hàng đợi, họ xếp hàng trả tiền trước lấy vé trước. Với ẩn dụ về hàng người đợi, thỉnh thoảng hàng đợi được gọi là danh sách chờ.

[43] **Xử lí giao dịch trực tuyến:** là chế độ xử lý mà 1 yêu cầu xử lý được thực hiện ngay lập tức và trả về kết quả, như hệ thống đặt chỗ của tàu hỏa và máy bay. Ví dụ, khi vé được yêu cầu, nó được in ra ngay lập tức. Một yêu cầu để xử lý gọi là 1 giao dịch .

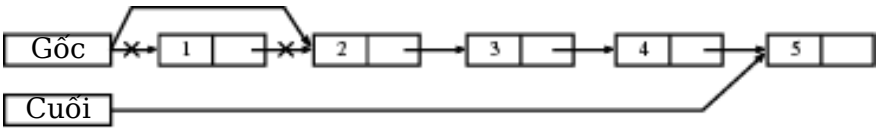
[44] (Gợi ý) Một hệ thống chia xẻ thời gian (TSS) có bề ngoại giống như một quá trình xử lí giao dịch trực tuyến, nhưng phương pháp xử lí hoàn toàn khác. 1 hàng đợi các công việc cần xử lý theo thứ tự đến của chúng, một TSS chia thời gian xử lý giữa các công việc. Thậm chí nếu 1 chương trình không kết thúc quá trình xử lý của nó, sau 1 thời gian, 1 chương trình khác sẽ bắt đầu quá trình xử lý. 1 TSS đã được hoàn thành bởi đa chương trình.

Chèn vào danh sách[45]

Hình vẽ dưới minh họa cách phần tử “5” được chèn vào. Giá trị con trỏ xác định “đuôi” cần được thay đổi trỏ vào “5”. Sau đó con trỏ của phần tử “4” vốn trước đó được sử dụng để trỏ vào phần tử cuối cùng, được thay đổi trỏ tới phần tử vừa được chèn vào.

Xóa khỏi danh sách

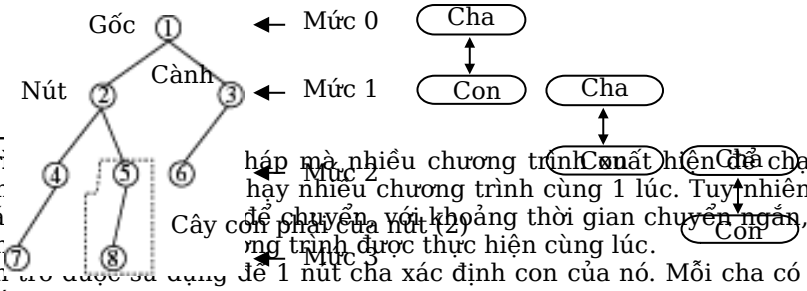
Khi xóa phần tử đầu tiên “1”, con trỏ gốc bị thay đổi trỏ tới phần tử “2”. Đọc phần tử “1” và xem giá trị con trỏ của nó, vốn trỏ tới vị trí của phần tử “2”.



1.3.5 Cây (Tree)

Điểm chính	➢ Cây xác định 1 cấu trúc phân lớp ➢ Trong các loại cây, cây nhị phân sẽ được tìm hiểu kĩ trong phần này .

Cây là một cấu trúc dữ liệu biểu diễn cấu trúc phân lớp giữa các phần tử. Nó được sử dụng cho biểu đồ tổ chức của một công ty, cấu hình hệ thống, … Nó có một **gốc** ở trên cùng, và các **nút** được liên kết bằng các **cành**. một nút ngay trên một nút khác được gọi là “**cha**” và một nút ở ngay dưới một nút khác được gọi là “**con**” [46]. Mỗi nút được nằm ở một mức (level) chỉ ra độ sâu, gốc nằm ở mức 0. Nút không có con gọi là “**lá**”. Một phần của cây được gọi là **cây con**. Cho một nút, cây con bên trái của nó được gọi là “**cây con trái**”, cây con ở bên phải gọi là “**cây con phải**”.



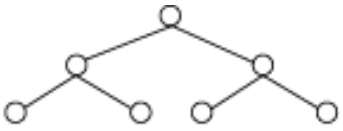
[45] **Đa chương tr** háp mà nhiều chương trình cùng xuất hiện để chạy cùng 1 lúc. Không máy tính r hay nhiều chương trình cùng 1 lúc. Tuy nhiên, máy tính xử dụng phương phá để chuyển, với khoảng thời gian chuyển ngắn, chương trình được thực hiện nh ng trình được thực hiện cùng lúc.

[46] (Gợi ý) Một con trỏ được sử dụng để 1 nút cha xác định con của nó. Mỗi cha có nhiều con trỏ tới các con của nó.

◆ Cây nhị phân và cây nhị phân hoàn chỉnh

Một cây mà mỗi nút có không quá 2 con được gọi là một "**cây nhị phân**". Nếu một cây nhị phân mà tất cả các lá có cùng độ sâu hoặc nếu 2 lá bất kì có độ sâu chênh lệch nhỏ hơn hoặc bằng một và các lá được đặt từ trái sang, được gọi là "**cây nhị phân hoàn chỉnh**"[47].

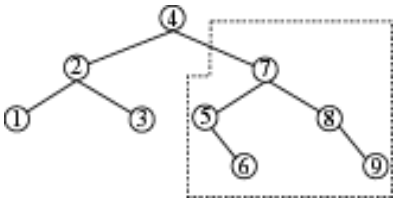
Cây nhị phân



Cây nhị phân hoàn chỉnh

◆ Cây nhị phân tìm kiếm

Một **cây nhị phân tìm kiếm** là một cây nhị phân mà giá trị của một phần tử được gán cho mỗi nút thỏa mã các ràng buộc[48]:
Giá trị con trái < giá trị phần tử cha < giá trị con phải



Giá trị của tất cả các nút đều lớn hơn giá trị (4) của gốc.

[47] (FAQ) Trong bài kiểm tra FE cơ bản, các câu hỏi chỉ liên quan tới cây nhị phân, hãy ghi nhớ chính xác đặc điểm của các loại cây nhị phân, như cây nhị phân đầy đủ, cây nhị phân tìm kiếm, đống.

[48] (Gợi ý) Trong 1 cây nhị phân tìm kiếm, chú ý rằng phần tử với giá trị nhỏ nhất là lá ngoài cùng bên trái trong khi phần tử có giá trị lớn nhất là lá ngoài cùng bên phải. Đây là một đặc điểm của cây nhị phân tìm kiếm.

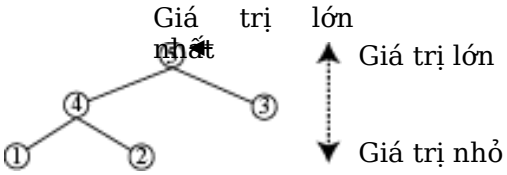
◆ Đống

một cây nhị phân được gọi là **đống** nếu giá trị của nút được gán từ mức gốc và từ trái sang phải trên cùng mức với điều kiện sau: [49]

Giá trị của phần tử cha > giá trị của phần tử con
(hoặc giá trị của phần tử cha < giá trị của phần tử con)

Đống phù hợp với điều kiện trên gọi là đống cực đại, và đống phù hợp với điều kiện dưới là đống cực tiểu.

Như một kết quả, phần tử có giá trị lớn (hoặc nhỏ) càng gần với nút, ngược lại phần tử có giá trị nhỏ (lớn) hướng tới lá. Nó là một cấu trúc dữ liệu phù hợp cho việc lấy giá trị lớn nhất (hoặc nhỏ nhất) với nút là phần tử có giá trị lớn nhất (hoặc nhỏ nhất).
.



1.3.6 Băm (Hash)

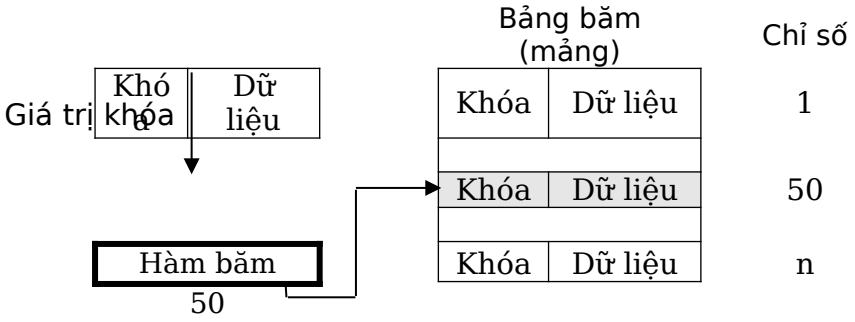
Điểm chính	➢ Băm là khái niệm sử dụng trực tiếp giá trị khóa như chỉ số ➢ Hai phương pháp để tránh xung đột là phương pháp địa chỉ mở và phương pháp dây chuyền.

Băm là khái niệm sử dụng trực tiếp giá trị khóa như vị trí lưu trữ dữ liệu. Ví dụ, có một mảng H kích thước 100 phần tử. Nếu giá trị khóa là các số có 2 chữ số từ 01 tới 99 không lặp lại, các giá trị khóa này có thể sử dụng như các chỉ số. Đó được gọi là **phương pháp tìm kiếm trực tiếp**.

Tuy nhiên, hiếm khi giá trị khóa được sử dụng trực tiếp như chỉ số. Vì vậy cần chuyển giá trị khóa sang chỉ số, **một hàm băm**[50] được sử dụng để tính giá trị băm, được sử dụng như chỉ số. Mảng lưu trữ các phần tử sử dụng phương pháp này gọi là **bảng băm**.

Xem như hàm băm là hàm chia giá trị khóa cho số phần tử của mảng và cộng thêm một vào phần dư.

[49] (Gợi ý) Chú ý rằng đống ở đây có giá trị lớn nhất nằm ở gốc. Lấy phần tử gốc, tái cấu trúc đống, và lặp lại quá trình xử lý này, với cách này, có thể lấy các phần tử theo thứ tự giá trị của chúng từ phần tử lớn nhất tớn phần tử nhỏ nhất.
[50] **Hàm băm**: 1 hàm tính địa chỉ dữ liệu (ví dụ: chỉ số) từ giá trị khóa.



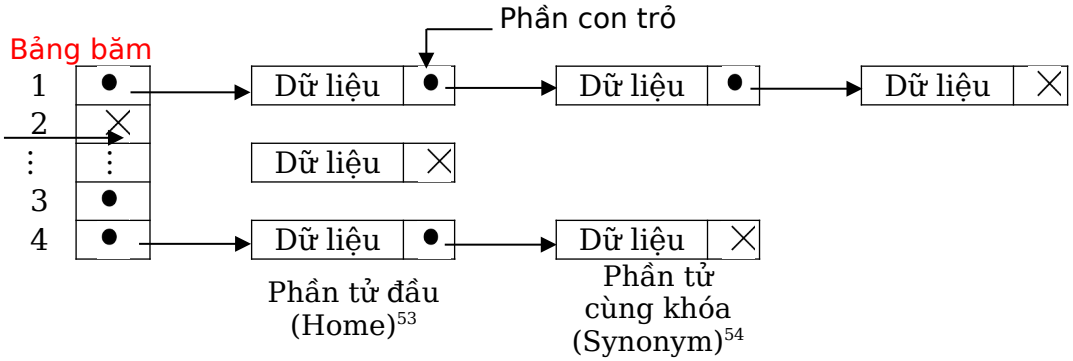
Nếu có n phần tử, số dư sẽ nằm từ 0 tới (n – 1), cộng 1 sẽ trả về giá trị băm từ 1 tới n. Chúng có thể được sử dụng như chỉ số được lưu trữ trong mảng.[51]

Tuy nhiên, khóa liên quan tới nhiều giá trị, chỉ số giống nhau có thể sinh ra từ giá trị khóa khác nhau bằng cách tính chỉ số (giá trị băm) từ 1 đến n sử dụng hàm băm. Khi giá trị băm giống nhau được sinh ra theo cách này được gọi là "**xung đột**"[52].

◆ Phương pháp dây chuyền (phương pháp băm mở)

Đây là phương pháp sử dụng một danh sách để lưu trữ phần tử có giá trị băm giống nhau khi xảy ra một xung đột. Bảng băm đầu tiên chỉ chứa con trỏ trỏ tới dữ liệu đầu tiên của danh sách.

Hình vẽ dưới minh họa một ví dụ có 3 mảnh dữ liệu được lưu trữ trong một vị trí có chỉ số là 1 trong bảng băm. Bảng băm này có một con trỏ xác định dữ liệu đầu tiên. Vị trí của dữ liệu tiếp theo được tìm thấy bởi lần theo phần con trỏ khi dữ liệu đầu tiên được đọc.

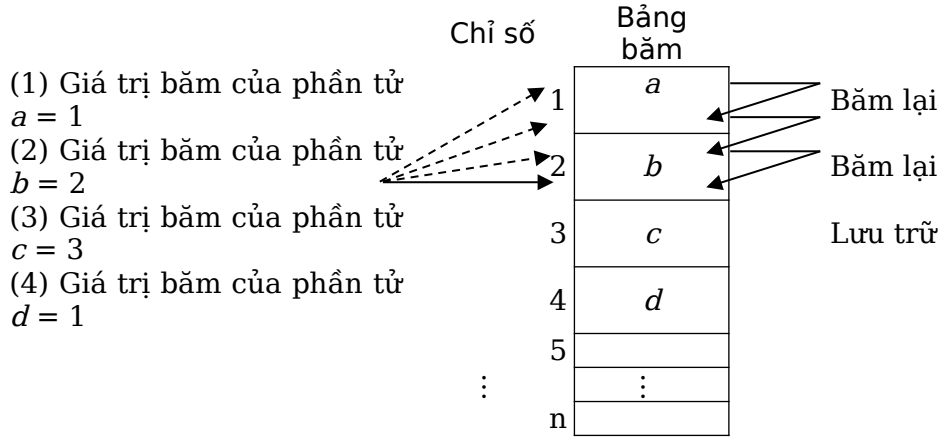


[51] (FAQ) Nhiều câu hỏi liên quan tới băm sẽ hỏi bạn tính vị trí lưu trữ và 1 hàm "mod" thường được sử dụng làm hàm băm. Trong đó "mod (*a,b*)" là lấy số dư trong phép chia "*a*" cho "*b*".
[52] **Xung đột**: Khi một hàm băm được sử dụng để tính địa chỉ lưu trữ, các khóa khác nhau có thể cho cùng 1 giá trị băm. Hiện tượng đó được gọi là xung đột.
[53] **Phần tử đầu (Home)**: Dữ liệu được lưu trữ đầu tiên khi xảy ra xung đột
[54] **Phần tử cùng khóa (Synonym)**: Dữ liệu đến sau khi xảy ra xung đột

◆ Phương pháp địa chỉ mở (phương pháp băm đóng)

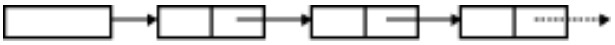
Đây là phương pháp giải quyết/ xử lý xung đột bằng băm lại. Băm lại là tính toán lại vị trí lưu trữ khi một xung đột xảy ra và lưu trữ dữ liệu mới ở đó nếu vị trí rỗng

Ví dụ, phần tử a, b, c được lưu trữ trong các vị trí riêng biệt (xác định bởi chỉ số) theo giá trị băm được tính toán. Tiếp theo, phần tử d có giá trị băm là 1, nhưng vị trí 1 đã bị chiếm bởi phần tử a. Ví dụ nếu phương pháp băm lại là "giá trị băm gốc + 1", thì vị trí tiếp theo được chỉ ra bởi chỉ số 2. Nhưng vị trí đó cũng bị chiếm, đi tiếp tới chỉ số 3. Tìm kiếm vị trí 4, vị trí đó trống. Như kết quả, phần tử d được lưu trữ ở vị trí có chỉ số 4. Nếu xảy ra trường hợp không có chỗ trống tới cuối của bảng băm, trở lại vị trí đầu tiên của bảng và tìm kiếm chỗ trống đầu tiên theo cách tương tự.



Câu hỏi nhanh

Q1 Ta gọi cấu trúc dữ liệu mà khái niệm của nó được biểu diễn trong hình dưới đây là gì?



Q2 Ta gọi cấu trúc dữ liệu kiểu "Last-In First-Out (Vào sau ra trước)" là gì?

Q3 Ta gọi cấu trúc dữ liệu kiểu "First-In First-Out (Vào trước ra trước)" là gì?

Q4 Định nghĩa "cây nhị phân" và "cây nhị phân hoàn chỉnh".

Q5 Ta gọi cây nhị phân thỏa mãn quan hệ sau là gì: "giá trị của con trái < giá trị của phần tử cha < giá trị con phải"?

A1 Danh sách

A2 Ngăn xếp

A3 Hàng đợi

A4 Cây nhị phân: Cây mà mỗi nút của nó có không quá 2 con
Cây nhị phân hoàn chỉnh: Cây nhị phân mà tất cả các lá có chiều sâu như nhau hoặc 2 lá bất kì có chênh lệch chiều sâu nhỏ hơn hoặc bằng 1 và các lá hướng về phía trái

A5 Cây nhị phân tìm kiếm

1.4 Giải thuật

Mở đầu

Một tập hợp các thủ tục để giải quyết một vấn đề gọi là một giải thuật. Hình vẽ biểu thị tập các thủ tục để đạt tới kết quả thích hợp gọi là lưu đồ. Chúng ta sẽ nghiên cứu một số giải thuật cơ bản.[55]

1.4.1 Các giải thuật tìm kiếm

Điểm chính	➢ Có 2 loại giải thuật tìm kiếm: tìm kiếm tuyến tính và tìm kiếm nhị phân ➢ Trong tìm kiếm nhị phân, phần tử được sắp xếp theo thứ tự tăng dần hoặc giảm dần

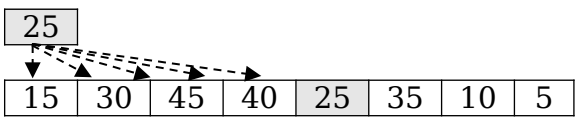
Tìm kiếm nghĩa là tìm một phần tử trong một bảng (mảng một chiều), và có 2 phương pháp tìm kiếm: tìm kiếm tuyến tính (tuần tự) và tìm kiếm nhị phân. Tìm kiếm tuần tự có thể thực hiện mà không quan tâm tới các phần tử được sắp xếp như thế nào, nhưng tìm kiếm nhị phân yêu cầu các phần tử đã được sắp xếp theo thứ tự tăng dần hoặc giảm dần

◆ Tìm kiếm tuyến tính (tuần tự)

Đây là phương pháp tìm kiếm các phần tử được yêu cầu trong bảng theo thứ tự từ vị trí đầu tiên của bảng. Nó có thể thực hiện mà không quan tâm tới các phần tử được sắp xếp như thế nào, nhưng nó lâu hơn tìm kiếm nhị phân. Nếu N là số phần tử, cần ít nhất 1 phép so sánh (nếu phần tử cần tìm nằm ở vị trí đầu của bảng) và nhiều nhất N phép so sánh (nếu phần tử cần tìm năm ở vị trí cuối cùng của bảng hoặc không tồn tại)

Trong tìm kiếm tuyến tính, phép so sánh được tạo ra từ chỉ số 1 và tiếp tục cộng 1 để tới chỉ số tiếp theo đến khi chỉ số bằng N.

Ví dụ, phần tử “25” trong bảng được tìm bằng phương pháp tìm kiếm tuyến tính. Nó được so sánh với giá trị đầu tiên, thứ 2, …, thứ 5. Các số xác định vị trí của phần tử là các chỉ số



[55] (FAQ) Để biểu diễn các giải thuật, các câu hỏi trong phần thi buổi sáng sử dụng các lưu đồ, trong khi các câu hỏi trong phần thi buổi chiều sử dụng giả ngôn ngữ. Các luật trong giả ngôn ngữ không được trình bày, hãy là đọc lướt nó

Chỉ số	1	2	3	4	5

◆ Tìm kiếm nhị phân

Đây là một phương pháp tìm kiếm hiệu quả khi các phần tử trong bảng được sắp xếp theo thứ tự tăng dần hoặc giảm dần. Phép so sánh được thực hiện tuần tự với giá trị nằm ở giữa của bảng. Sau phép so sánh đầu tiên, nửa bên phải hoặc bên trái của bảng bị vứt bỏ, và giá trị ở giữa của phần còn lại được sử dụng cho phép so sánh tiếp theo. Phạm vi để tìm kiếm giảm một nửa sau mỗi lần, kiểu tìm kiếm này nhanh hơn thời gian trung bình của tìm kiếm tuyến tính

Giải thích một giải thuật cụ thể, sử dụng mảng sau làm ví dụ, coi như chúng ta tìm kiếm giá trị "11"

Chỉ số	1	2	3	4	5	6	7	8	9	10
Mảng *T*	0	1	3	5	7	9	11	13	15	17

Lần so sánh đầu tiên

Phạm vi là toàn bộ mảng. Đặt L là giới hạn dưới và U là giới hạn trên của phạm vi. Đặt M là giá trị ở giữa (trung vị)

Chỉ số	1	2	3	4	5	6	7	8	9	10
Mảng T	0	1	3	5	7	9	11	13	15	17

L ← Khoảng tìm kiếm → *U*

$M = (L + U) / 2 = (1 + 10) / 2 = 5.5 \rightarrow 5$ (Trung vị)

Trung vị có thể tính bằng cách làm tròn trên hoặc làm tròn dưới đều được. Ở đây ta chọn cách làm tròn dưới.

$T(M) = T(5) = 7$

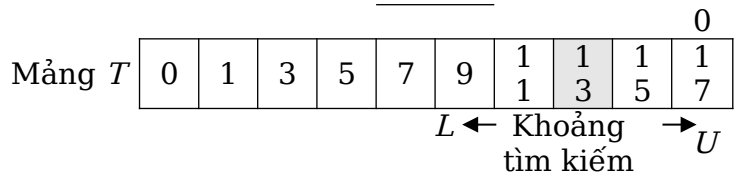
Giá trị cần tìm là "11" nên "11" không thể được tìm thấy ở nửa trái của bảng chứa giá trị trung vị bởi vì các phần tử được sắp xếp theo thứ tự tăng dần và giá trị cần tìm lớn hơn giá trị trung vị[56].

Lần so sánh thứ hai

Lần so sánh thứ nhất chỉ rõ ràng rằng giá trị cần tìm không nằm trong nửa trái của bảng chứa giá trị trung vị, ta thay đổi khoảng tìm kiếm. Ở đây, giới hạn dưới bị thay đổi tới giá trị ngay bên phải trung vị. Giá trị của L bị thay đổi như sau:
L = M + 1 = 5 + 1 = 6
Kết quả, khoảng tìm kiếm thay đổi như sau

Chỉ số	1	2	3	4	5	6	7	8	9	1

[56] (Gợi ý) Khi xóa nửa bên trái của bảng, chỉ số dưới mới bằng trung vị cộng 1; khi xóa nửa bên phải, chỉ số trên mới bằng trung vị trừ 1



Tương tự như phép so sánh đầu tiên, chúng ta tìm trung vị mới như sau:

$M = (L + U) / 2 = (6 + 10) / 2 = 8$ (Trung vị)

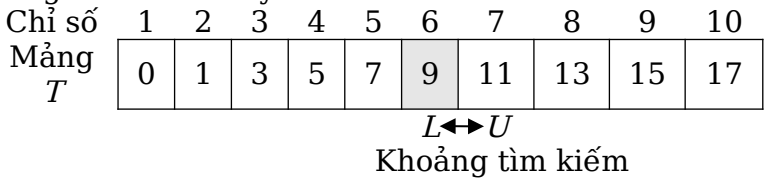
$T(M) = T(8) = 13$
Ta sẽ so sánh nó với "11", giá trị cần tìm "11" không thể nằm ở nửa bên phải của khoảng tìm kiếm chứa giá trị trung vị mới. Do các phần tử được sắp xếp theo thứ tự tăng dần và giá trị cần tìm nhỏ hơn giá trị trung vị

Lần so sánh thứ ba

Lần so sánh thứ hai chỉ ra rằng giá trị cần tìm không nằm trong nửa bên phải của khoảng tìm kiếm chứa trung vị, chúng ta sẽ thay đổi khoảng tìm kiếm. Ở đây, giới hạn trên bị thay đổi tới giá trị ngay bên phải của trung vị. Giá trị của U bị thay đổi như sau:

$U = M - 1 = 8 - 1 = 7$

Kết quả, khoảng tìm kiếm thay đổi như sau



Tương tự như lần so sánh thứ 2, ta tìm trung vị mới như sau:

$M = (L + U) / 2 = (6 + 7) / 2 = 6$ (Trung vị)

$T(M) = T(6) = 9$

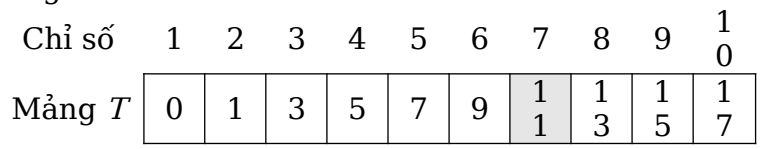
Ta sẽ so sánh nó với "11", giá trị cần tìm "11" không thể nằm ở nửa trái của khoảng tìm kiếm chứa giá trị trung vị

Lần so sánh thứ tư

Lần so sánh thứ ba chỉ rõ rằng giá trị cần tìm không nằm trong nửa trái của khoảng tìm kiếm chứa giá trị trung vị, ta sẽ thay đổi giới hạn dưới giống như lần so sánh thứ hai

$L = M + 1 = 6 + 1 = 7$

Kết quả, khoảng tìm kiếm như sau



$L = U$
Khoảng tìm kiếm

$M = (L + U) / 2 = (7 + 7) / 2 = 7$ (trung vị)

$T(M) = T(7) = 11$
Ta sẽ so sánh nó với "11", ta tìm thấy giá trị cần tìm[57]

Thủ tục khi tìm kiếm gặp lỗi

Giả sử ta cần tìm "10", trong lần so sánh thứ tư, công thức "T(M) = 11 > 10" là đúng, ta phải thay đổi giới hạn trên của khoảng tìm kiếm. Khoảng tìm kiếm mới như sau:

$L = 7$ (Không đổi)
$U = M - 1 = 7 - 1 = 6$

Vì L là giới hạn dưới và U là giới hạn trên, nên " $L \leq M$," nhưng bây giờ có " L > M". Khi bất đẳng thức này xảy ra, ta kết luận rằng phần tử cần tìm không tồn tại.

◆ So sánh giữa phương pháp tìm kiếm tuyến tính và tìm kiếm nhị phân

Khi sử dụng phương pháp tìm kiếm nhị phân để tìm "11", cần 4 phép so sánh. Tuy nhiên, trong phương pháp tìm kiếm tuyến tính, do "11" có chỉ số là "7", nó cần 7 phép so sánh. Do đó, phương pháp tìm kiếm nhị phân phức tạp hơn nhưng số phép so sánh giảm xuống

Tuy nhiên, xét trường hợp tìm "0", chỉ số của nó là "1". Phương pháp tìm kiếm tuyến tính có thể tìm thấy kết quả ở ngay lần so sánh đầu tiên trong khi phương pháp tìm kiếm nhị phân cần 3 phép so sánh. Ở đây tìm kiếm nhị phân nhanh hơn.

Để đánh giá vấn đề này, có một khái niệm gọi là độ phức tạp tính toán. Khi số phần tử N là rất lớn, giá trị này nói cho ta biết rằng trung bình cần bao nhiêu phép so sánh. Ta bỏ qua giải thích chi tiết ở đây, nhưng thừa nhận công thức sau:

Độ phức tạp của phương pháp tìm kiếm tuyến tính = N / 2
Số phép so sánh tối đa của phương pháp tìm kiếm tuyến tính = N
Độ phức tạp của phương pháp tìm kiếm nhị phân = $[\log_2 N]$
Số phép so sánh tối đa của phương pháp tìm kiếm nhị phân = số phép so sánh trung bình + 1

Ta sử dụng $\log_2 N$ trong ngoặc vuông. Nhìn chung, $\log_2 N$ không phải là một số nguyên, nhưng số phép so sánh phải là một số nguyên. Do vậy, [ ] biểu thị xóa, hoặc cắt phần thập phân, ví dụ [10.513] là 10

[57] (Gợi ý) Phần tử "11" là T(7) trong mảng T, được tìm thấy sau 4 phép so sánh trong tìm kiếm nhị phân. Tìm kiếm tuyến tính cần 7 phép so sánh để tìm thấy giá trị này

1.4.2 Các giải thuật sắp xếp

Điểm chính	➢ Cẩn thận khi điều khiển chỉ số trong phương pháp sắp xếp nổi bọt, sắp xếp chọn, sắp xếp chèn ➢ Lời gọi đệ qui được sử dụng trong sắp xếp nhanh và sắp xếp trộn

Sắp xếp nghĩa là bố trị lại các phần tử và/hoặc các bản ghi của một mảng theo thứ tự của khóa. Sắp xếp các phần tử từ khóa nhỏ nhất tới khóa lớn nhất được gọi là "sắp xếp theo thứ tự tăng dần" và sắp xếp theo chiều từ khóa lớn nhất tới khóa nhỏ nhất gọi là "sắp xếp theo thứ tự giảm dần".

Sắp xếp nội dung của 1 vùng trong 1 chương trình, ví dụ 1 mảng, được gọi là "sắp xếp trong" trong khi sắp xếp dữ liệu được lưu trữ ở thiết bị ngoại vi như các bản ghi trong 1 file, được gọi là "sắp xếp ngoài" (sắp xếp file). Các phương pháp sắp xếp điển hình của sắp xếp trong gồm: sắp xếp nổi bọt, sắp xếp chọn, sắp xếp chèn, sắp xếp nhanh, sắp xếp trộn, sắp xếp vỏ bọc, sắp xếp đống

◆ Sắp xếp nổi bọt

Trong sắp xếp nổi bọt, mỗi cặp phần tử liền nhau được so sánh tuần tự và đổi chỗ nếu cần thiết. Trong trường hợp sắp xếp theo thứ tự tăng dần, giá trị lớn nhất được đặt là phần tử cuối cùng của mảng. Tiếp theo, trở lại từ đầu, các giá trị được kiểm tra và thay đổi khi cần thiết. Trong lần chạy thứ hai, phần tử ở vị trí cuối cùng của mảng bị đặt ra ngoài khoảng sắp xếp. Tiếp tục quá trình này, khoảng sắp xếp nhỏ dần sau mỗi lần, và kết thúc sắp xếp khi phần tử đầu tiên và phần tử thứ hai được so sánh

Dưới đây là ví dụ của sắp xếp theo thứ tự tăng dần

Trước khi sắp xếp	5		4		3	2	1:	
Lần 1	5		4		3	2	1:	Đổi chỗ 5 và 4
	4	⇔	5		3	2	1:	Đổi chỗ 5 và 3
	4		3	⇔	5	2	1:	Đổi chỗ 5 và 2
	4		3		2	⇔ 5	1:	Đổi chỗ 5 và 1
	4		3		2	1 ⇔	5:	Kết thúc lần 1 (giá trị lớn nhất nằm cuối cùng bên phải)
Lần 2	4	⇔	3		2	1 |	5:	Đổi chỗ 4 và 3 (giá trị bên phải của "|" đã được sắp xếp)
	3		4	⇔	2	1 |	5:	Đổi chỗ 4 và 2
	3		2		4	⇔1 |	5:	Đổi chỗ 4 và 1
	3		2		1	4 |	5:	Kết thúc lần 2 (giá trị lớn thứ hai nằm ở vị trí thứ hai từ bên phải)
Lần 3	3	⇔	2		1 |	4	5:	Đổi chỗ 3 và 2 (giá trị bên phải của "|" đã được sắp xếp)
	2		3	⇔	1 |	4	5:	Đổi chỗ 1 và 3

	2	1	3	4	5:	Kết thúc lần 3 (giá trị lớn thứ ba nằm ở vị trí thứ ba từ bên phải)
Lần 4	2	1 |	3	4	5:	Đổi chỗ 2 và 1 (giá trị bên phải của "|" đã được sắp xếp)
	1	2 |	3	4	5:	Kết thúc lần 4, sắp xếp xong

◆ Sắp xếp chọn

Phương pháp sắp xếp chọn tìm giá trị lớn nhất (hoặc nhỏ nhất) từ mảng và thay đổi nó với phần tử nằm ở cuối mảng. Tiếp theo, tìm phần giá trị lớn nhất (hoặc nhỏ nhất) từ mảng trừ phần tử cuối cùng và đổi vị trí của nó với phần tử thứ hai từ cuối lên của mảng. Lặp lại thủ tục này, sắp xếp chọn kết thúc khi nó so sánh phần tử đầu tiên và phần tử thứ hai của mảng[58]

Dưới đây là ví dụ sắp xếp theo thứ tự tăng dần

Lần 1	5 4 3 2 1:	Vì "5" là giá trị lớn nhất, nó đổi chỗ với phần tử cuối cùng "1".
Lần 2	1 4 3 2 | 5:	Vì 4 là giá trị lớn nhất, nó đổi chỗ cho phần tử cuối cùng trong lần chạy thứ 2.
Lần 3	1 2 3 | 4 5:	Sắp xếp xong

◆ Sắp xếp chèn

Phương pháp sắp xếp chèn bắt đầu với 1 mảng đã được sắp xếp, so sánh phần tử cần chèn với các phần tử trong mảng từ đầu tới cuối, và chèn phần tử vào vị trí thích hợp[59]. Ví dụ dưới, các phần tử bên trái "|" đã được sắp xếp, ở đây chỉ có duy nhất 1 phần tử trong lần chạy đầu tiên. Nó coi như đã được sắp xếp.

Dưới đây là 1 ví dụ về sắp xếp theo thứ tự tăng dần

Lần 1	5 | 4 3 2 1:	Vì 4 là giá trị nhỏ nhất, nó được chèn vào vị trí thích hợp (trước 5)
Lần 2	4 5 | 3 2 1:	Vì 3 là giá trị nhỏ nhất, nó được chèn vào vị trí thích hợp (trước 4)
Lần 3	3 4 5 | 2 1:	Vì 2 là giá trị nhỏ nhất, nó được chèn vào vị trí thích hợp (trước 3)
Lần 4	2 3 4 5 | 1:	Vì 1 là giá trị nhỏ nhất, nó được chèn vào vị trí thích hợp (trước 2)
	1 2 3 4 5:	Sẵp xếp xong

[58] (FAQ) Sắp xếp nổi bọt và sắp xếp chọn thường xuyên xuất hiện trong bài thi. Các câu hỏi được đưa ra theo nhiều cách, ví dụ như nội dung của mảng ở trạng trạng thái trung gian và điền vào chỗ trống trong lưu đồ. Hãy chắc rằng bạn đã hiểu rõ giải thuật

[59] (Gợi ý) Khi tìm kiếm 1 vị trí để chèn trong sắp xếp chèn, phần tử được chèn được so sánh từ cuối của mảng đã được sắp xếp. Ví dụ, trong lần chạy thứ 3, thứ tự so sánh sẽ là "2 và 5," "2 và 4," "2 và 3"

◆ Sắp xếp nhanh

Sắp xếp nhanh lựa chọn 1 giá trị ngẫu nhiên từ mảng và sử dụng nó như 1 chốt, các phần tử được chia thành 2 nhóm, nhóm đầu tiên chứa tất cả các phần tử nhỏ hơn chốt và nhóm thứ hai chứa tất cả các phần tử lớn hơn chốt (giá trị bằng có thể đặt ở 1 trong 2 nhóm). Tiếp theo, thủ tục tương tự được lặp đi lặp lại cho mỗi nhóm. Nó tiếp tục cho đến khi chỉ còn duy nhất 1 phần tử trong mỗi nhóm. Kết quả mảng đã được sắp xếp[60]

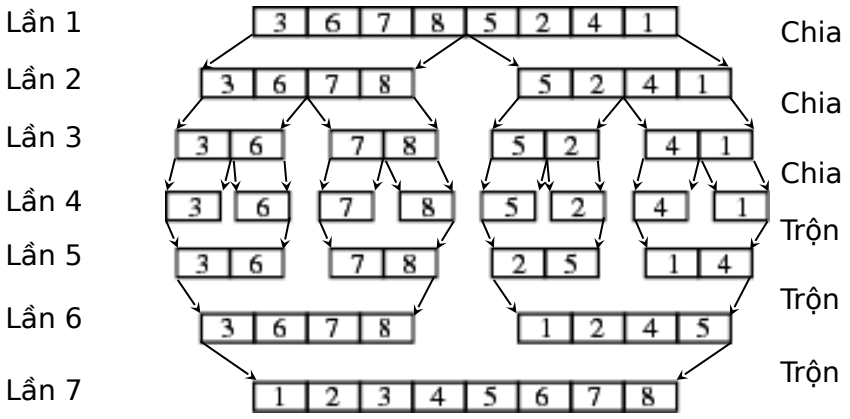
Dưới đây là 1 ví dụ của sắp xếp theo thứ tự tăng dần. Các giá trị gạch chân là chốt. Đường "|" xác định ranh giới của khối.

Lần 1	2	5	6	4	1	3:	Chia thành 2 khối
Lần 2	2	1	3	5	6	4:	Chia mỗi khối làm 2
			|			
Lần 3	1	2	3	4	5	6:	Chia mỗi khối làm 2 (trừ các nhóm chỉ có 1 phần tử)
		|	|	|			
Lần 4	1	2	3	4	5	6:	Sắp xếp xong
	|	|	|	|	|		

[60] (Chú ý) Sắp xếp nhanh và sắp xếp trộn khác nhau ở số phần tử liên quan trong quá trình chia, nhưng chúng sử dụng cùng 1 phương pháp. Trong trường hợp này, phương pháp "gọi đệ quy" được sử dụng

◆ Sắp xếp trộn

Trong phương pháp sắp xếp trọn, 2 hoặc nhiều mảng, mỗi mảng đều đã được sắp xếp, được trộn với nhau thành 1 mảng đã sắp xếp. Trong sắp xếp trộn, phép chia được lặp lại đến khi mỗi nhóm chỉ có duy nhất 1 phần tử. Khi mỗi nhóm chỉ có duy nhất 1 phần tử, các phần tử được trộn vào nhau tuần tự[61].
Dưới đây là 1 ví dụ sắp xếp theo thứ tự tăng dần



◆ Sắp xếp bằng phương pháp Shellsort

Đây là một dạng cải tiến của phương pháp sắp xếp chọn, sắp xếp được làm nhanh hơn bởi tăng khoảng cách di chuyển của các phần tử

Đầu tiên, các phần tử được sắp xếp hỗn độn băng phương pháp chèn với khoảng trống 1 kích thước nào đó. Sau đó, sắp xếp chèn được sử dụng lại để hoàn thành thao tác sắp xếp

Dưới đây là ví dụ của sắp xếp theo thứ thự tăng dần. Khởi đầu, khoảng trống được thiết lập là 2, và chỉ sắp xếp với các phần tử đó. Sau đó, khoảng trống là 1 và sắp xếp chèn được sử dụng

Chưa sắp xếp	2	4	5	3	1:	
Lần đầu	2	4	5	3	1:	Mọi phần tử khác được sắp xếp (các phần tử gạch chân đã được sắp xếp)
	1	4	2	3	5:	Chạy xong lần đầu
Lần 2	1	4	2	3	5:	Mọi phần tử còn lại được sắp xếp (các phần tử gạch chân đã được sắp xếp)
	1	3	2	4	5:	Chạy xong lần hai
Lần 3	1	2	3	4	5:	Chạy xong lần 3 (sắp xếp xong)

Lý do để một phương pháp phức tạp như vậy được sử dụng là sắp sếp chèn không cần yêu cầu thay đổi các phần tử. Ví dụ, xét tình huống sau
Case *A*: 2 4 6 | 1 ...

[61] **Đệ quy**: Là quá trình mà 1 hàm gọi chính nó trong bản thân nó. Trong Pascal và C "gọi đệ quy" được phép nhưng COBOL và Fortran không cho phép điều này.

Case *B*: 2 4 6 | 8 …

Trong trường hợp A, để quyết định nơi chèn "1", so sánh với 6 → 4 → 2. Sau đó, tất cả các phần tử cần được di chuyển để tạo khoảng trống để chèn "1". Ngược lại, trong trường hợp B, ngay lập tức gái trị được so sánh với "6", vị trí chèn được xác định mà không cần dịch chuyển

Vì vậy, tổng số xử lý trong phương pháp chèn dựa trên các phần tử được sắp xếp như thế nào đầu tiên. Sắp xếp bóc vỏ giảm việc di chuyển các phần tử bởi lần sắp xếp phác đầu tiên

◆ Sắp xếp vun đống

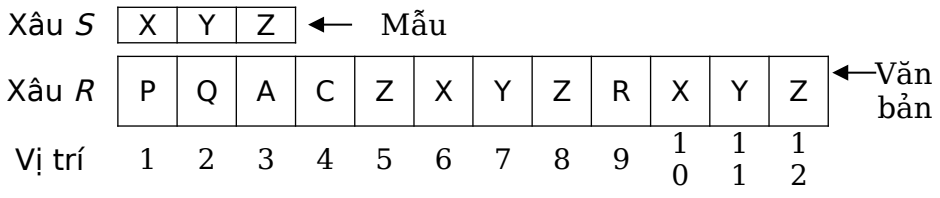
1 đống là 1 cây nhị phân trong đó mọi cây con có tính chất: phần tử cha có giá trị lớn hơn các con của nó. Nếu phần tử gốc được chọn, ta có thể thu được giá trị lớn nhất trong khi các phần tử còn lại có thể tái cấu trúc thành đống mới. Ta có thể lấy lại gốc, mà đưa cho ta phần tử có giá trị lớn thứ hai. Nói cách khác, bằng cách lặp lại thao tác lấy gốc và tái cấu trúc đống, sắp xếp có thể hoàn tất. Phương pháp sắp xếp sử dụng đống gọi là sắp xếp đống[62]

1.4.3 Các giải thuật tìm kiếm xâu

Điểm chính	➢ Nhìn chung, tìm kiếm xâu so sánh từng ký tự một. ➢ Các phương pháp tìm kiếm sâu gồm: phương pháp duyệt toàn bộ, phương pháp Boyer-Moore…

Tìm kiếm xâu là quá trình tìm một chuối kí tự đặc biệt trong một văn bản (xâu kí tự). Trong đa số trường hợp, xâu nằm trong mảng mà mỗi ô chứa 1 kí tự và được tham chiếu bởi chỉ số. Hai mảng được cho trước: văn bản và xâu tìm kiếm. Giải thuật sẽ tìm kiếm xâu mẫu trong xâu văn bản.

Trong ví dụ dưới, chúng ta muốn kiểm tra xâu S là "XYZ", có mặt ở ô 6~8 và ô 10~12 trong xâu R. Rất rõ ràng khi quan sát bằng mắt, nhưng trên thực tế rất khó để viết giải thuật để kiểm tra



[62] (FAQ) Với sắp xếp nhanh, sắp xếp trộn, sắp xếp chèn, sắp xếp đống, sắp xếp bóc vỏ, các câu hỏi thường liên quan tới khái niêm của chúng, bạn cần hiểu quá trình sắp xếp dữ liệu như thế nào

◆ Phương pháp duyệt toàn bộ

Phương pháp duyệt toàn bộ (tìm kiếm vét cạn) là phương pháp trong đó xâu cần tìm được bằng cách so sánh từng kí tự từ đầu dãy. Đây là phương pháp giống với tìm kiếm tuyến tính. Tìm kiếm kết thúc khi kí tự cuối cùng của dãy được so sánh với kí tự cuối cùng của xâu cần tìm kiếm. Ví dụ sau giải thích cụ thể cho phương pháp trên[63]

Văn bản	P	Q	A	B	C	Z	X	Y	Z	R	X	Y	Z
Mẫu	X	Y	Z										

(1) Kí tự đầu tiên của mẫu được so sánh với kí tự đầu tiên của văn bản

Văn bản	[P]	Q	A	B	C	Z	X	Y	Z	R	X	Y	Z
Mẫu	[X]	Y	Z										

(2) Do không trùng khớp, kí tự thứ hai của văn bản được so sánh với kí tự đầu tiên của mẫu

Văn bản	P	[Q]	A	B	C	Z	X	Y	Z	R	X	Y	Z
Mẫu	[X]	Y	Z										

(3) Lặp lại quá trình này và đến kí tự thứ 7 “X” xảy ra trùng khớp

Văn bản	P	Q	A	B	C	Z	[X]	Y	Z	R	X	Y	Z
Mẫu							[X]	Y	Z				

(4) Vì có 1 sự tương ứng, tiếp theo kí tự thứ 8 của văn bản được so sánh với kí tự thứ hai của mẫu

Văn bản	P	Q	A	B	C	Z	X	[Y]	Z	R	X	Y	Z
Mẫu							X	[Y]	Z				

(5) Vì cặp thứ hai cũng trùng khớp, kí tự thứ ba được so sánh

Văn bản	P	Q	A	B	C	Z	X	Y	[Z]	R	X	Y	Z
Mẫu							X	Y	[Z]				

Bây giờ chúng ta xác nhận rằng mẫu kí tự S nằm trong xâu kí tự R

[63] (Gợi ý) Trong tìm kiếm xâu, cần 1 chỉ số cho xâu S và 1 chỉ số khác cho xâu R. Khi trả lời câu hỏi, điểm cốt yếu là nắm vững cách làm thế nào sử dụng các chỉ số

◆ Phương pháp Boyer-Moore

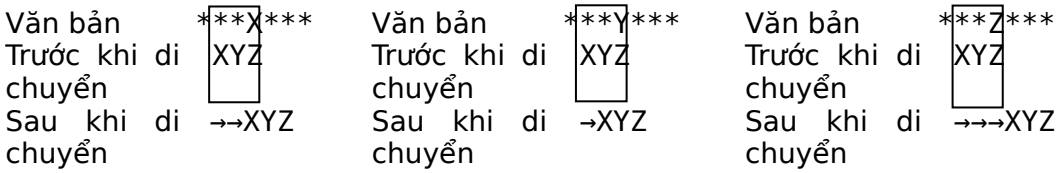
Đây là một phương pháp lấy nội dung của xâu mẫu trong văn bản bỏ qua những phần vô giá trị. Nếu mẫu và 1 xâu của văn bản không trùng khớp, số kí tự có thể nhảy qua dựa trên kí tự ngoài cùng bên phải của khoảng tìm kiếm của văn bản được so sánh

Ta giải thích chi tiết dùng ví dụ như trong tìm kiếm vét cạn

(1) Nếu kí tự ngoài cùng bên phải của phần văn bản đang được so sánh với chuỗi là “X”, vị trí tiếp theo có khả năng mà mẫu có thể trùng là sau đó 2 kí tự, nên 2 kí tự tiếp theo được nhảy qua

(2) Nếu kí tự ngoài cùng bên phải của phần văn bản đang được so sánh với chuỗi là “Y”, vị trí tiếp theo có khả năng mà mẫu có thể trùng là sau đó 1 kí tự, nên kí tự tiếp theo được nhảy qua

(3) Nếu kí tự ngoài cùng bên phải của phần văn bản đang được so sánh với chuỗi là “Z”, vị trí tiếp theo có khả năng mà mẫu có thể trùng là sau đó 3 kí tự, nên 3 kí tự tiếp theo được nhảy qua


Văn bản ***X***
Trước khi di chuyển XYZ
Sau khi di chuyển →→XYZ
Văn bản ***Y***
Trước khi di chuyển XYZ
Sau khi di chuyển →XYZ
Văn bản ***Z***
Trước khi di chuyển XYZ
Sau khi di chuyển →→→XYZ

(4) Nếu kí tự ngoài cùng bên phải của phần văn bản không phải là X, Y hoặc Z thì tình huống giống hệt với (3), nên 3 kí tự tiếp theo được nhảy qua[64]

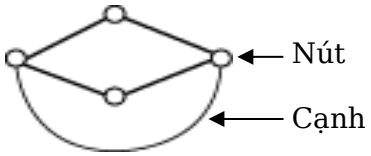
[64] (Chú ý) Trong phương pháp BM, cần tăng tính toán số kí tự được nhảy qua. Ví dụ đã thảo luận ở đây có mẫu 3 kí tự, nên số là 2 cho X, 1 cho Y và 3 cho Z hoặc bất kì kí tự nào khác. Những điều này cần được tính toán trước khi bắt đầu tìm kiếm xâu

1.4.4 Các giải thuật trên đồ thị

Điểm chính	➢ Cây là một dạng đồ thị. ➢ Thứ tự trên cây tìm kiếm có thể là ưu tiên chiều rộng hoặc ưu tiên chiều sâu

Một giải thuật trên đồ thị là một giải thuật mà phép tìm kiếm được thực hiện trên một cây, một cấu trúc dữ liệu hướng câu hỏi. Dựa trên thứ tự tìm kiếm, giải thuật trên đồ thị có thể là ưu tiên chiều rộng hoặc ưu tiên chiều sâu. Giải thuật ưu tiên chiều sâu thường xuất hiện trong đề thi, do đó phải đảm bảo rằng bạn hiểu rõ cách lấy các nút trong giải thuật này,

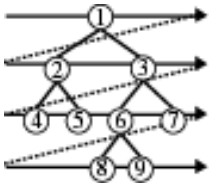
Một đồ thị được tạo thành bởi các nút và cạnh. Một nút là một đỉnh của đồ thị và cạnh là đoạn nối giữa 2 đỉnh. Đây là ví dụ về 1 đồ thị.[65]



Một cây có thể coi là một đồ thị trong đó không phải tất cả các nút được nối với tất cả các nút khác.

◆ Thứ tự ưu tiên chiều rộng

Phép tìm kiếm bắt đầu ở gốc và duyệt ngang qua các nút ở mức thấp từ trái sang phải. Trong ví dụ dưới đây, số ở các nút xác định thứ tự mà nút đó được duyệt qua.



[65] (Gợi ý) Khi nghe thuật ngữ đồ thị, bạn nghĩ tới 1 biểu đồ hình tròn, 1 biểu đồ cột... nhưng trong thế giới toán học nó nói tới 1 tập các đỉnh và cạnh

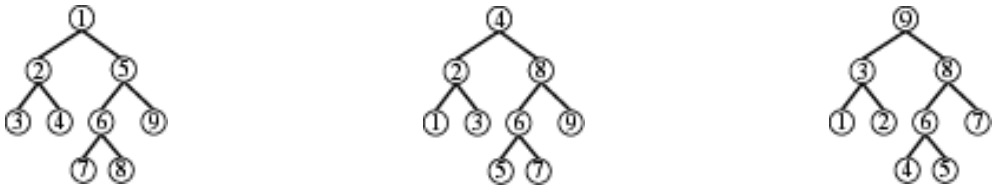
◆ Thứ tự ưu tiên chiều sâu

Trong tìm kiếm ưu tiên chiều sâu[66], chúng ta bắt đầu ở gốc, duyệt từ cây con trái rồi từ lá. Dựa trên thời điểm mà các nút được duyệt, ta có thể chia lớp như bảng dưới

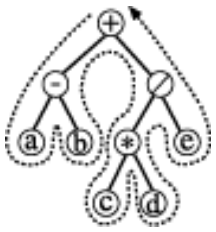
Phương pháp duyệt	Thứ tự duyệt các nút
Duyệt trước	Thứ tự: cha, con trái, con phải
Duyệt giữa	Thứ tự: con trái, cha, con phải
Duyệt sau	Thứ tự: con trái, con phải, cha

Trong hình dưới, số trên các nút xác định thứ tự mà nút được duyệt

Duyệt trước Duyệt giữa Duyệt sau



Các luật cho mỗi phép tìm kiếm có thể chưa thật sự rõ ràng, chúng ta sẽ đưa thêm một số giải thích. Trong phép tìm kiếm ưu tiên chiều sâu, thứ tự tìm kiếm như hình dưới



Trong thứ tự duyệt trước, giá trị của các nút được lấy ra khi duyệt qua bên trái của nút đó. Vì thế,thứ tự sẽ là "+ - a b / * c d e". Trong thứ tự duyệt giữa, giá trị các nút được lấy ra khi duyệt qua bên dưới của nút. Vì thế thứ tự sẽ là "a – b + c * d / e". Trong thứ tự duyệt sau, giá trị mỗi nút được truy cập khi duyệt qua bên phải của nút. Do đó, thứ tự là "a b – c d * e / +".[67]

[66] (FAQ) Câu hỏi duyệt theo chiều sâu thường xuất hiện trong bài thi, hãy hiểu rõ các nút được lấy ra như thế nào trong thứ tự duyệt trước, duyệt giữa, duyệt sau.

[67] (Chú ý) Chú ý kết quả phép duyệt cây để lấy các kí hiệu và các biến. Trong duyệt trước, kết quả là "+ – a b / * c d e", chính là kí pháp Ba Lan. Trong duyệt giữa, kết quả là "a – b + c * d / e", là dạng công thức toán học chuẩn. Trong duyệt sau, kết quả là "a b – c d * e / +" chính là dạng kí pháp Ba Lan ngược.

Câu hỏi nhanh

Q1 Trong tìm kiếm nhị phân, khi số lượng dữ liệu đã sắp xếp tăng gấp 4 lần thì số lượng phép so sánh tối đa tăng bao nhiêu?

Q2 Trình bày đặc điểm của các phương pháp sắp xếp: “shell short”, “bubble sort,” “quick sort,” and “heap sort.”

A1

2 lần

Khi số lượng dữ liệu tăng 4 lần, thay thế “n” trong công thức “$\log_2 n + 1$” số lượng phép so sánh tối đa bằng “$4n$.”

$$\begin{aligned}\log_2 4n + 1 &= (\log_2 4 + \log_2 n) + 1 \\ &= \log_2 2^2 + \log_2 n + 1 \\ &= 2 + \log_2 n + 1 \\ &= 2 + (\log_2 n + 1)\end{aligned}$$

A2

Sắp xếp Shellsort: Phần tử của mảng được lấy ở chính giữa và được sắp xếp, sau đó các phần tử được lấy bằng cách giảm khoảng cách và được sắp xếp.

Sắp xếp nổi bọt: Các phần tử liền nhau được so sánh và đổi chỗ nếu không đúng thứ tự, quá trình này được lặp đi lặp lại.

Sắp xếp nhanh: Chọn một giá trị (ở giữa) làm giá trị tham chiếu, mảng được chia thành tập các phần tử lớn hơn và tập các phần tử nhỏ hơn giá trị tham chiếu. Với mỗi phần, quá trình trên được lặp đi lặp lại.

Sắp xếp đống: Phần chưa được sắp xếp được biểu diễn dưới dạng cây con, từ đó giá trị lớn nhất (hoặc nhỏ nhất) được lấy ra và đưa vào phần đã được sắp xếp. Quá trình này được lặp đi lặp lại để giảm dần phần chưa được sắp xếp

Câu hỏi 1

Độ khó: ** Tần suất: ***

Q1. Có 1 thanh ghi nhị phân. Sau khi nhập 1 số nguyên dương x vào thanh ghi này, thao tác "dịch thanh ghi sang trái 2 bit rồi cộng thêm x" được thực hiện. Hỏi giá trị thanh ghi gấp mấy lần x. Giả thiết không xảy ra tràn trong quá trình dịch.

a) 3 b) 4 c) 5 d) 6

Đáp án câu 1

Đáp án đúng: c

Tổng quát, nếu không bị tràn, phép dịch n bit sang trái nhân giá trị đó với 2^n trong khi phép dịch n bit sang phải nhân giá trị với $1/2^n$. Dịch sang trái 2 bit là nhân với 2^2, nếu y là kết quả của phép tính, y liên hệ với x bằng phương trình.

$$\begin{aligned} y &= x\times 2^2 + x \\ &= x\times(2^2+1) \\ &= 5\times x \text{ (y is 5 times x)} \end{aligned}$$

a) Để tạo số lớn gấp 3 lần x, ta chỉ cần dịch giá trị thanh ghi sang trái 1 bit và cộng thêm x. Dịch 1 bit sang trái nhân giá trị với 2^1, kết quả sẽ như sau

$$y = x\times 2^1 + x = 2x + x = 3x$$

b) Để tạo số lớn gấp 4 lần, chúng ta dịch thanh ghi sang trái 2 bit, giá trị ban đầu sẽ được nhân với 2^2. Kết quả sẽ là

$$y = x\times 2^2 = 4x$$

d) Để tạo số lớn gấp 6 lần, ta dịch thanh ghi giá trị sang trái 2 bit và cộng giá trị này với giá trị thu được bằng cách dịch thanh ghi giá trị ban đầu sang trái 1 bit. Dịch 2 bit sang trái nhân giá trị với 2^2 và dịch sang trái 1 bit nhân giá trị với 2^1, nên kết quả sẽ là

$$y = x\times 2^2 + x\times 2^1 = 4x + 2x = 6x$$

Câu hỏi 2

Độ khó: * Tần suất: ***

Q2. Mô tả nào sau đây thích hợp với triệt tiêu chữ số có nghĩa.

a) Nó nghĩa là số chữ số có nghĩa suy giảm trầm trọng khi một số dấu phẩy động bị trừ bởi số khác gần bằng nó
b) Nó tham chiếu tới một lỗi xảy ra bởi kết quả phép tính vượt quá giá trị số lớn nhất có thể xử lý
c) Nó tham chiếu tới một lỗi xảy ra khi làm tròn (trên hoặc dưới) số nhỏ hơn chữ số có nghĩa thấp nhất khi số lượng chữ số trong biểu diễn số học bị giới hạn
c) Nó tham chiếu tới bỏ quên chữ số có trọng số thấp trong 1 phép toán khi cộng các số dấu phẩy động

Đáp án câu 2

Đáp án đúng: a

Triệt tiêu chữ số có nghĩa là hiện tượng các chữ số có nghĩa bị mất trong phép trừ 2 giá trị cùng dấu gần bằng nhau và trong phép cộng 2 giá trị ngược dấu có giá trị tuyệt đối gần bằng nhau. Nó xảy ra do máy tính chỉ xử lý các số có hữu hạn chữ số. Ví dụ, nó xảy ra trong phép tính sau:

```
  123.4567
– 123.4556
  0.0011
```

Ở đây, chữ số có nghĩa cao nhất bằng 0, số chữ số có nghĩa suy giảm nghiêm trọng

b) Mô tả tràn số

c) Mô tả lỗi làm tròn

d) Mô tả mất chữ số đuôi

Câu hỏi 3

Độ khó: *** Tần suất: ***

Q3. Bảng chân lý dưới đây là kết quả của phép toán logic "x @ y". Biểu diễn nào sau đây là tương đương với phép toán này

x	*y*	*x* @ *y*
True	True	False
True	False	False
False	True	True
False	False	False

a) *x* OR (NOT *y*)
b) (NOT *x*) AND *y*
c) (NOT *x*) AND (NOT *y*)
d) (NOT *x*) OR (NOT y)

Đáp án câu 3

Đáp án đúng: b

Trong các phép toán logic, ta gán "1" cho "đúng" và "0" cho "sai". Dễ dàng hơn nếu sử dụng kí pháp quen thuộc, ta sẽ sử dụng các kí hiệu sau

x AND y → $x \cdot y$ (tích logic)
x OR y → $x + y$ (tổng logic)
NOT x → $\overline{x}$ (phủ định logic)

Vậy, biểu thức logic trong đáp án có thể viết lại như sau:

a) x OR (NOT y) $x + \overline{y}$
b) (NOT x) AND y $\overline{x} \cdot y$
c) (NOT x) AND (NOT y) $\overline{x} \cdot \overline{y}$
d) (NOT x) OR (NOT y) $\overline{x} + \overline{y}$

Ta kiểm tra để xem biểu thức nào trong nhóm đáp án trùng khớp (có kết quả giống) với phép toán logic được đưa ra:

x	y	$\overline{x}$	$\overline{y}$	a) $x + \overline{y}$	b) $\overline{x} \cdot y$	c) $\overline{x} \cdot \overline{y}$	d) $\overline{x} + \overline{y}$	*x* @ *y*
1	1	0	0	1	0	0	0	0
1	0	0	1	1	0	0	1	0
0	1	1	0	0	1	0	1	1
0	0	1	1	1	0	1	1	0

Do đó, phép toán có kết quả trùng với x @ y là $\overline{x} \cdot y$.

Câu hỏi 4

Độ khó: ** Tần suất: **

Q4. Cú pháp cho các <giá trị số> được định nghĩa như sau, biểu thức nào sau đây xem như
<giá trị số>?

<giá trị số> ::= <chuỗi số>|<chuỗi số>E<chuỗi số>|
<chuỗi số>E<dấu><chuỗi số>
<chuỗi số> ::= <chữ số>|<chuỗi chữ số> <chữ số>
<chữ số> ::= 0|1|2|3|4|5|6|7|8|9
<dấu> ::= +|-

a) –12 b) 12E–10 c +12E–10 d) +12E10

Đáp án câu 4

Đáp án đúng: b

Câu trả lời tuân theo dạng thứ 3 (<chuỗi số> E <dấu> <chuỗi số>) của định nghĩa <giá trị số>.

Kiểu định nghĩa này được gọi là kí pháp BNF. Kí pháp BNF được sử dụng như 1 cách biểu diễn chính thức cú pháp của 1 ngôn ngữ lập trình

Tổng quan kí pháp BNF như sau:

α::=β → Kí hiệu bên tay trái α được định nghĩa bởi kí hiệu bên tay phải β. Nói cách khác, α = β.

< α > → Kí hiệu này biểu thị biến α. <> có thể bỏ qua
| → Nghĩa là "hoặc" "α::=β | γ" nghĩa là "α::=β" or "γ.".

"::=" có thể viết đơn giản là "="

a) Trong định nghĩa <giá trị số>, "–" (<dấu>) phải nằm sau "E". Phần gạch chân không thỏa mãn định nghĩa –12

c) Trong định nghĩa <giá trị số>, "+" (<dấu>) phải nằm sau "E". Phần gạch chân không thỏa mãn định nghĩa ±12E – 10

Trong định nghĩa <giá trị số>, "+" (<dấu>) phải nằm sau "E". Phần gạch chân không thỏa mãn định nghĩa ±12E10

Câu hỏi 5

Độ khó: ** Tần suất: **

Q5. 1 khóa được tạo bởi 3 kí tự alphabe. Khi giá trị băm h được quyết định với biểu thức sau, khóa nào sẽ xung đột với khóa "SEP"? Ở đây "a mod b" biểu diễn số dư khi a chia cho b

h = (Tổng vị trí của các chữ cái) mod 27

Bảng chữ cái	Vị trí
A	1
B	2
C	3
D	4
E	5
F	6
G	7
H	8
I	9
J	10
K	11
L	12
M	13

Bảng chữ cái	Vị trí
N	14
O	15
P	16
Q	17
R	18
S	19
T	20
U	21
V	22
W	23
X	24
Y	25
Z	26

a) APR b) FEB c) JAN d) NOV

Đáp án câu 5

Đáp án đúng: b

Một khóa băm là kết quả biến đổi khóa bởi 1 hàm băm, được sử dụng cho quá trình băm. Thuật ngữ "hashing- quá trình băm/giải thuật băm" nói tới 1 quá trình thực hiện các phép tính trên khóa để biến đổi nó thành 1 giá trị địa chỉ để lấy địa chỉ lưu trữ của bản ghi trong file. Ở đây hàm được sử dụng để lấy địa chỉ được gọi là 1 hàm băm. Nếu giải thuật băm sinh ra giá trị băm giống nhau cho hoặc nhiều khóa khác nhau, điều đó gọi là xung đột. Các bản ghi ở sau khi 1 xung đột xảy ra gọi là đồng nghĩa. Tính giá trị băm cho "SEP" bằng hàm băm đã cho, ta thu được kết quả sau:

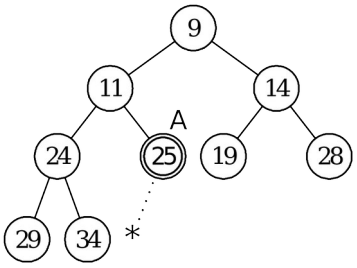
h = (tổng vị trí của mỗi chữ cái được sử dụng trong khóa) mod 27
= (19 + 5 + 16) mod 27
= (40) mod 27
= 13 (40 ÷ 27 = 1 dư 13)

a) "ARP" (1 + 18 + 16) mod 27 = 8 (35 ÷ 27 = 1 dư 8)
b) "FEB" (6 + 5 + 2) mod 27 = 13 (13 ÷ 27 = 0 dư 13) — Đụng độ
c) "JAN" (10 + 1 + 14) mod 27 = 25 (25 ÷ 27 = 0 dư 25)
d) "NOV" (14 + 15 + 22) mod 27 = 24 (51 ÷ 27 = 1 dư 24)

Câu hỏi 6

Độ khó: ** Tần suất: **

Q6. Cho đống dưới đây, giá trị của nút cha nhỏ hơn giá trị của các nút con. Khi chèn 1 nút vào đống này, 1 phần tử được thêm vào ở cuối cùng. Nếu phần tử đó nhỏ hơn nút cha, cha và con phải chuyển chỗ cho nhau. Nếu phần tử 7 được thêm vào đống ở vị trí được đánh dấu (*), phần tử nào sẽ nằm ở vị trí A.

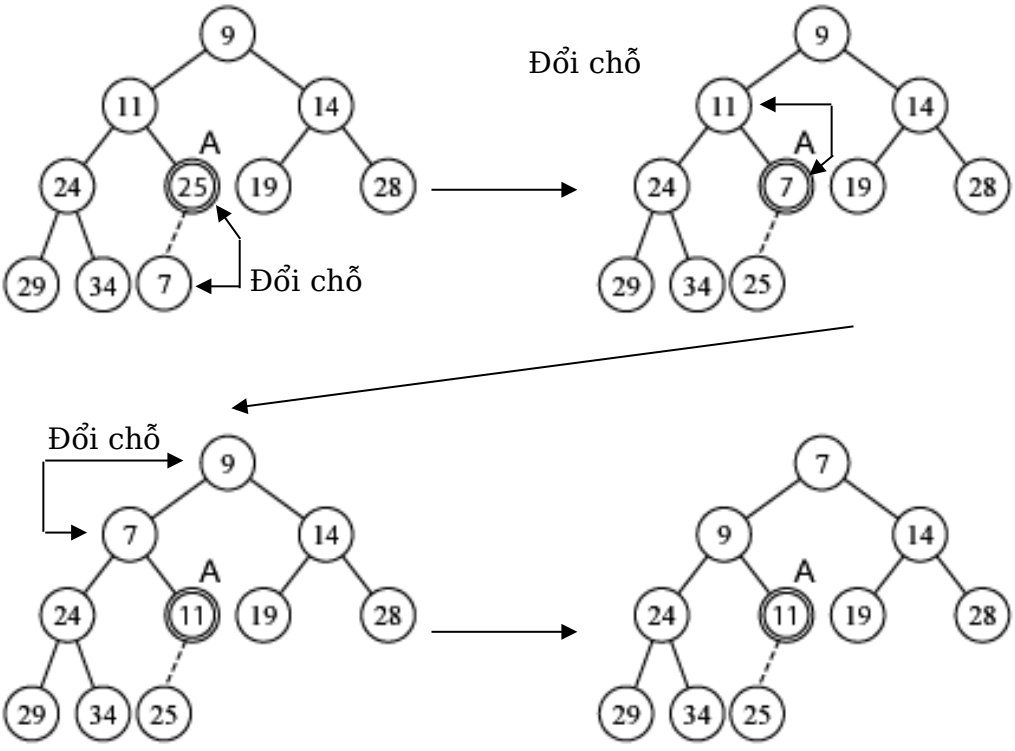


a) 7 b) 11 c) 24 d) 25

Đáp án câu 6

Đáp án đúng: b

Thêm phần tử vào vị trí đã cho sau đó lặp lại thủ tục chuyển chỗ cha và con khi phần tử con có giá trị nhỏ hơn giá trị của cha. “7” được thêm vào ở đây



Bây giờ, đống đã hoàn thành. Do đó phần tử ở vị trí A ( ) là “11”

Câu hỏi 7

Độ khó: * Tần suất: ***

Q7. Thuật ngữ nào sau đây biểu diễn đặc điểm của các thao tác trên ngăn xếp

a) FIFO b) LIFO c) LILO d) LRU

Đáp án câu 7

Đáp án đúng: b

Ngăn xếp là 1 cấu trúc dữ liệu dạng "Last-In First-Out", với dữ liệu được lưu trữ cuối cùng sẽ được lấy ra đầu tiên. Thao tác chèn dữ liệu vào ngăn xếp được gọi là "push" và thao tác lấy dữ liệu khỏi ngăn xếp được gọi là "pop"

a) FIFO là cấu trúc dữ liệu của hàng đợi với dữ liệu được lưu trữ đầu tiên sẽ được lấy ra đầu tiên

b) LILO (LInux LOader) là 1 bộ nạp khởi động (chương trình để nạp hệ điều hành vào bộ nhớ) cho phép PCs hiểu Linux

c) LRU nghĩa là được truy cập ít nhất trong giai đoạn gần đây và được sử dụng như 1 giải thuật thay trang trong hệ thống bộ nhớ ảo. Đây là phương pháp thay trang bỏ đi trang được truy cập ít nhất

Câu hỏi 8

Độ khó: ** Tần suất: **

Q8. Bảng quyết định dưới đây để tạo các báo cáo từ tệp nhân viên. Điều gì dưới đây có thể kết luận từ bảng quyết định này

Dưới 30 tuổi	Y	Y	N	N
Nam	Y	N	Y	N
Đã kết hôn	N	Y	Y	N
Xuất ra báo cáo 1	-	X	-	-
Xuất ra báo cáo 2	-	-	-	X
Xuất ra báo cáo 3	X	-	-	-
Xuất ra báo cáo 4	-	-	X	-

a) Báo cáo 1 chứa nội dung của Báo cáo 4 trừ dữ liệu của nam từ 30 tuổi trở

lên

b) Báo cáo 2 chứa tất cả nam chưa lấy vợ

c) Nam trong báo cáo 3 nằm trong báo cáo 2

d) Những người trong báo cáo 4 không nằm trong bất kì báo cáo nào khác

Đáp án câu 8

Đáp án đúng: d

Có phủ định của "kết hôn" là "chưa kết hôn" và phủ định của "nam" là "nữ." Đọc mô tả về đáp án cẩn thận. Trong các giải thích dưới đây, phần gạch chân là phủ định (N)

a) Điều kiện thỏa mãn cho báo cáo 1 là "dưới 30, không phải nam, đã kết hôn" → "dưới 30, nữ, đã kết hôn"

Điều kiện thỏa mãn cho báo cáo 4 là "không dưới 30, nam, đã kết hôn" → "tối thiểu 30, nam, đã kết hôn"

Nên, báo cáo 1 chỉ chứa nữ. Báo cáo 4 chỉ chứa nam và bỏ đi điều kiện "nam, tối thiểu 30" ở Report 4 làm nó trở thành tập rỗng. Do vậy, mô tả này là sai

b) Điều kiện thỏa mãn báo cáo 2 là "không dưới 30, không phải nam, chưa kết hôn" → "tối thiểu 30, nữ, chưa kết hôn"

Nên báo cáo 2 chỉ chứa nữ, nó không đúng với mệnh đề "tất cả nam chưa kết hôn". Vậy mô tả này là sai

c) Điều kiện thỏa mãn báo cáo 3 là "dưới 30, nam, chưa kết hôn"

Điều kiện thỏa mãn báo cáo 2 là "không dưới 30, không phải nam, chưa kết hôn"

Nên, báo cáo 3 chứa chỉ chứa nam trong khi báo cáo 2 chỉ chứa nữ. Mô tả này là sai

d) Bằng cách loại trừ, đây phải là câu trả lời đúng, nhưng chúng ta sẽ kiểm tra nó. Tổ chức tất cả các tiêu chuẩn của tất cả các báo cáo ở câu a, b, c trong đáp án, ta có:

Báo cáo 1: "dưới 30, nữ, đã kết hôn" (từ "a")
Báo cáo 2: "tối thiểu 30, nữ, chưa kết hôn" (từ "b")
Báo cáo 3: "dưới 30, nam, chưa kết hôn" (từ "c")
Báo cáo 4: "tối thiểu 30, nam, đã kết hôn" (từ "a")

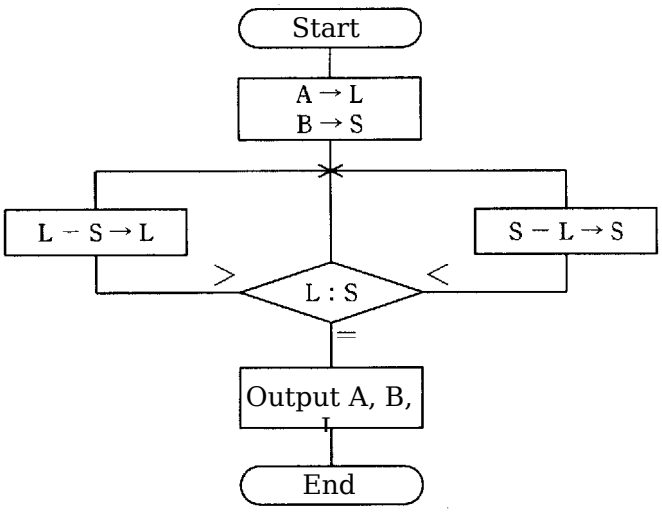
Điều kiện "tối thiểu 30" của báo cáo 4 cũng đúng với báo cáo 2, nhưng

điều kiện khác là phủ định của nhau, nên không người nào nằm trong cả 2. Ngoài ra điều kiện “nam” đúng với báo cáo 3, nhưng điều kiện khác là phủ định của nhau. Tương tự, điều kiện “đã kết hôn” đúng cho báo cáo 2, nhưng điều kiện khác là phủ định của nhau. Nên, không có báo cáo nào chứa người trong báo cáo 4. Đây là mô tả đúng

Câu hỏi 9

Độ khó: ** Tần suất: ***

Q9. Lưu đồ dưới minh họa giải thuật Euclide tìm ước chung lớn nhất của A và B bằng các lặp lại phép trừ. Khi A là 876 và B là 204, có bao nhiêu phép so sánh được thực hiện trong quá trình này



a) 4 b) 9 c) 10 d) 11

Đáp án câu 9

Đáp án đúng: d

Giải thuật Euclide là giải thuật tìm ước số chung lớn nhất của 2 số nguyên A và B. Tuy vậy bạn không cần hiểu giải thuật này, chỉ cần hiểu dữ liệu thay đổi như thế nào. Đầu tiên gán "A→L" và "B→S," giá trị của ước chung lớn nhất đạt được là L và S. Giải thuật quyết định đâu là số lớn hơn và trừ nó cho số nhỏ hơn. Khi "L=S" giải thuật kết thúc

Với khởi tạo A=876 và B=204, ta trừ B (=S) từ A (=L) nhiều lần. Chú ý rằng giá trị cần được so sánh trước khi trừ

(1) Với điều kiện L=876 và S=204, lặp lại phép trừ đến khi L<S. Do $876 \div 204 = 4$ dư 60, phép trừ và thay thế "L – S → L" được thực hiện 4 lần trước khi "L < S" thỏa mãn. Do đó so sánh (L, S) xảy ra 4 lần

(2) Với điều kiện L=60 và S=204, lặp lại phép trừ đến khi L>S. Do $204 \div 60 = 3$ dư 24, phép trừ và thay thế "S – L → S" được thực hiện 3 lần trước khi "L > S" thỏa mãn. Do đó so sánh (L, S) xảy ra 3 lần

(3) Với điều kiện L=60 và S=24, lặp lại phép trừ đến khi L<S. Do $60 \div 24 = 2$ dư 12, phép trừ và thay thế "L – S → L" được thực hiện 2 lần trước khi "L < S" thỏa mãn. Do đó so sánh (L, S) xảy ra 2 lần

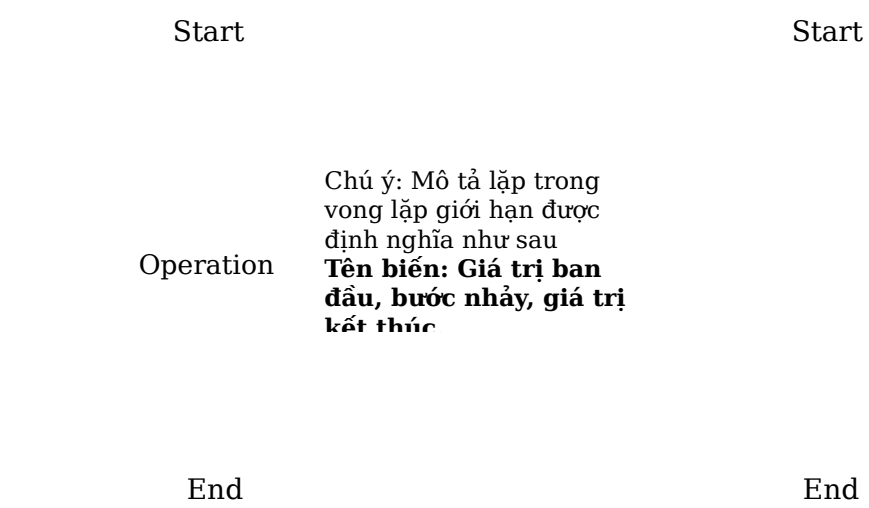
(4) Với điều kiện L=12 và S=24, lặp lại phép trừ đến khi L=S. Do $24 \div 12 = 2$ dư 0, phép trừ và thay thế "S – L → S" được thực hiện 2 lần trước khi "L = S" thỏa mãn. Do đó so sánh (L, S) xảy ra 2 lần

(5) Số lần phép so sánh "L:S" được tính như sau. Ta có thể cộng các số từ (1) đến (4). Tổng số phép so sánh = 4 + 3 + 2 + 2 = 11 lần

Câu hỏi 10

Độ khó: *** Tần suất: ***

Q10. Các giải thuật mô tả bằng 2 lưu đồ dưới đây được thực hiện trên 1 số nguyên dương M, điều kiện nào cần chèn thêm vào ô bên dưới sao cho nhận được cùng kết quả x?


Thao tác
Start
Start
Operation
Chú ý: Mô tả lặp trong vong lặp giới hạn được định nghĩa như sau
Tên biến: Giá trị ban đầu, bước nhảy, giá trị kết thúc
End
End

a) $n > M$ b) $n > M + 1$ c) $n > M - 1$ d) $n < M$

Đáp án câu 10

Đáp án đúng: a

Kí pháp "n: M, -1, 1" ở vòng lặp có nghĩa, như giải thích trong câu hỏi, như sau: lấy giá trị khởi tạo của n là M và trừ 1 mỗi lần lặp và dừng khi đạt giá trị "1". Điều này nghĩa là giá trị của n thay đổi từ M, M-1, …, 2, 1

Hãy xem xét lưu đồ bên trái trước. Bắt đầu với n = M, và giảm giá trị đi 1 trong mỗi vòng lặp đến khi nhận giá trị là 1 (n = M, M – 1, … , 2, 1), thao tác sau " $X \times n \rightarrow X$" được thực hiện bên trong vòng lặp. Khi "1→ x", giá trị khởi tạo của x là 1. Ta hãy theo dõi giá trị của x thay đổi như thế nào khi n thay đổi

$n=M$: $x\times n=1\times M=M\rightarrow x=M$ (Giá trị của x đổi thành M.)
$n=M-1$: $x\times n=M\times(M-1)=M(M-1)\rightarrow x=M(M-1)$
(Giá trị của x đổi thành $M(M-1)$.)
$n=M-2$:
$x\times n=M(M-1)\times(M-2)=M(M-1)(M-2)\rightarrow x=M(M-1)(M-2)$
(Giá trị của x đổi thành $M(M-1)(M-2)$.)

Và tiếp tục...

$n=2$: $x\times n=M(M-1)(M-2)..2\rightarrow x=M(M-1)(M-2)..2$
$n=1$: $x\times n=M(M-1)(M-2)..2\cdot1\rightarrow x=M(M-1)(M-2)..2\cdot1=M!$ (M giai thừa)

Kết quả tính $M\cdot(M-1)\cdot(M-2)\cdot\ldots\cdot2\cdot1$. Với 1 giá trị nguyên M, tích $M\cdot(M-1)\cdot(M-2)\cdot\ldots\cdot2\cdot1$ được gọi là $M!$ (M giai thừa)

Xem xét lưu đồ bên phải. n thay đổi từ 1, 2, ..., M và phá trình "$x\times n\rightarrow x$" được lặp trong vòng lặp. Lưu đò tính giai thừa bắt đầu từ 1.

Hãy theo dõi xem x thay đổi như thế nào với đối với giá trị n. Như trong lưu đồ bên trái, giá trị khởi tạo của x là 1. Hơn nữam mỗi lần vòng lặp được thực hiện, giá trị của n tăng thêm 1. Phần gạch chân trong mỗi dòng dưới là giá trị tiếp theo của x

$n=1$: $x\times n=\underline{1}\times1=1\rightarrow x$ ($x=1$)
$n=2$: $x\times n=\underline{1}\times2\rightarrow x$ ($x=1\times2$)
$n=3$: $x\times n=\underline{1\times2}\times3\rightarrow x$ ($x=1\times2\times3$)
$n=4$: $x\times n=\underline{1\times2\times3}\times4\rightarrow x$ ($x=1\times2\times3\times4$)

Hãy xem n lớn bao nhiêu để tạo kết quả giống như kết quả của lưu đồ bên trái. Lưu đồ bên trái lặp "$x\times n\rightarrow x$" để thực hiện tính toán dưới
Lưu đồ bên trái = $M\cdot(M-1)\cdot(M-2)\cdot\ldots\cdot2\cdot1$

Ở đây phép nhân bắt đầu với M; ở bên phải, phép nhân bắt đầu với 1. Do đó, nếu phép nhân tiếp tục đến M, kết quả của 2 lưu đồ sẽ bằng nhau
Lưu đồ bên phải = $1\times2\times3\times\ldots\times M$

Thế nên ta phải lặp "$x\times n\rightarrow x$" đến khi n = M. Theo lưu đồ, sau lệnh "$x\times n\rightarrow x$", chương trình thực hiện "$n+1\rightarrow n$", nên sau n = M được nhân, ta sẽ có n = (M+1). Điều này nghĩa là chương trình cần kết thúc khi "$n=M+1$" xảy ra. Các lựa chọn trong nhóm đáp án, điều kiện này là "$n>M$."